

"Restricted Materials of IBM"

All Rights Reserved

Licensed Materials - Property of IBM

©Copyright IBM Corp. 1982, 1986

LY28-1298-2

File No. S370-37

Program Product

**MVS/Extended Architecture
Interactive Problem
Control System
Logic and Diagnosis**

MVS/System Products:

JES2 Version 2 5740-XC6

JES3 Version 2 5665-291

IBM

Third Edition (June, 1986)

This is a major revision of LY28-1298-1. See the Summary of Amendments following the Contents for a summary of the changes made to this manual. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition applies to Version 2 Release 1.7 of MVS/System Product (5665-291 and 5740-XC6) and to all subsequent releases until otherwise indicated in new editions or Technical Newsletters. The previous edition still applies to Version 2 Release 1.2 and Version 2 Release 1.3 of MVS/System Product (5665-291 and 5740-XC6) and may be ordered using the temporary order number LQ68-1298. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department D58, Building 921-2, PO Box 390, Poughkeepsie, N.Y. 12602. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

This manual describes the internal logic of the MVS/XA interactive problem control system (IPCS). The information presented here is intended for readers who maintain IPCS and who need to know about IPCS' design, organization, and data areas.

Organization of This Book

This book is divided into the following sections:

Section 1: Introduction contains introductory information about IPCS, discussing its purpose, functional components, relationship to the operating system, and task and data structures.

Section 2: Method of Operation describes the utility commands and the five major functions of IPCS.

Section 3: Program Organization presents descriptions of the modules that perform the major functions of IPCS.

Section 4: Data Areas contains the data areas that IPCS references.

Note: For load module structure and directory information, refer to your installation's control data set (CDS).

Section 5: Diagnostic Aids contains information such as return codes and user completion ABEND codes that can be useful in the diagnosis of IPCS problems.

Required Reading

To understand the logic of IPCS, you should be familiar with its use, as described in the following publication:

MVS/Extended Architecture Interactive Problem Control System (IPCS) User's Guide and Reference, GC28-1297

Related Publications

Other publications that might be helpful are:

MVS/Extended Architecture Data Administration Guide, GC26-3783

MVS/Extended Architecture Integrated Catalog Administration: Access Method Services Reference, GC26-4019

MVS/Extended Architecture Interactive Problem Control System Reference Summary, SX28-0014

MVS/Extended Architecture Message Library: System Messages, Volume 1, GC28-1376, and Volume 2, GC28-1377

MVS/Extended Architecture System Programming Library: System Macros and Facilities, Volume 1 GC28-1150, Volume 2 GC28-1151

MVS/Extended Architecture System Programming Library: Service Aids,
GC28-1159

*MVS/Extended Architecture TSO Guide to Writing a Terminal Monitor Program
or a Command Processor,* GC28-1295

MVS/Extended Architecture System Programming Library: User Exits,
GC28-1147

*MVS/Extended Architecture VSAM Catalog Administration: Access Method
Services Reference,* GC26-4075

Contents

Section 1. Introduction	1
Purpose	1
Relationship to the Operating System	1
Task Structure	1
Master Task	3
Processing Subtask	3
ISPF Logical Screen Task	3
Data Structure	4
Control Blocks	4
Direct Access Data Sets	4
Functional Components	6
Initialization and Termination	8
Monitor	8
IPCS Subcommand Processors	8
ISPF Dialog Programs	9
IPCS Common Service Functions	9
Data Access Services	9
Print and Terminal Services	10
Message Text Build and Routing	10
Common Exit Services	10
Dump Access	10
Dump Initialization	11
Storage Map Management	11
Scan Routines	11
Symbol Management	11
Find Routines	11
Address Space Mapping	11
Address Resolution	11
Dump Directory Handling	11
Dump Data Formatting	12
Section 2. Method of Operation	13
TSO Utility Commands	13
SYSDSCAN Command	13
IPCSDDIR Command	14
IPCS Command	14
Initialization	16
Intertask Communication	17
Termination	17
Processing Subtask Termination	18
Master Task Termination	18
Monitor	19
Subcommand Monitoring	20
Subcommand Selection	20
Terminal Attention Processing	21
General Subcommand Processing	22
Related Subcommands	24
Storage Management	24
Program Management	25
IPCS Subcommand Processing	26
IPCS Session Control Commands and Subcommands	27
SETDEF Subcommand	27
HELP Subcommand	28
TSO Subcommand	28
END Subcommand	29
COPYDUMP Subcommand	29
Problem and Data Management Subcommands	32
ADDPROB Subcommand	32
DELPROB Subcommand	34
LISTPROB Subcommand	36
MODPROB Subcommand	39
ADDDSN Subcommand	41
DELDSN Subcommand	45
LISTDSN Subcommand	50
MODDSN Subcommand	52

Analysis Subcommands	55
ASMCHECK Subcommand	55
COMCHECK Subcommand	57
ENQCHECK Subcommand	59
STATUS Subcommand	61
SUMMARY Subcommand	63
PRDMP Exit Support Subcommands	64
Line-Oriented Dump Viewing Processing	66
FIND Subcommand	66
FINDMOD Subcommand	67
FINDUCB Subcommand	67
LIST Subcommand	67
RUNCHAIN Subcommand	68
Full-Screen Dump Viewing Processing (DSPL3270)	69
Full-Screen Initialization	69
Screen Image Generation	70
Response Processing	70
Termination	71
CLIST Support Subcommands	77
COMPARE Subcommand	77
EVALUATE Subcommand	77
EVALDEF Subcommand	78
EVALDUMP Subcommand	78
EVALMAP Subcommand	78
EVALSYM Subcommand	78
INTEGER Subcommand	78
ISPEXEC Subcommand	79
NOTE Subcommand	79
Dump Directory Handling Subcommands	79
DROPDUMP Subcommand	79
LISTDUMP Subcommand	80
LISTSYM and DROPSYM Subcommands	80
EQUATE Subcommand	81
RENUM Subcommand	81
STACK Subcommand	81
LISTMAP, DROPMAP, and SCAN Subcommands	81
Parse Validity Checking	82
ISPF Dialog Programs	83
Logical Screen Task Initialization	83
Processing IPCS Procedures	83
Dump Viewing Dialog	83
Common Service Functions	87
Data Access Services	87
Data Access Services Overview	87
Invoking Data Access Services	88
Data Set Allocation	89
Data Set Access Requests	92
Data Set Deallocation	96
Print and Terminal Services	98
Message Handling	98
Routing Routines (Routers)	99
Transmitting Routines	99
Message Text Build and Routing	99
Common Exit Services	100
Exit Services Router	100
Control Block Formatting Service	101
Format Model Processor Service	102
Select ASID Service	104
ECT Exit Services	106
Get Symbol Service	107
Equate Symbol Service	107
Dump Access	108
Dump Initialization	109
Open Dump	109
Closed Dump	109
Sequential Initialization	110
Open Processing	111
Direct Initialization	111
Storage Map Management	112

The Validation Process	114
Scan Routines	114
A Hypothetical Scan Routine	114
Phases of Scan Routine Execution	115
Types of Tests	116
Symbol Management	116
Finding Data Areas	119
The Find Process	119
Operation of a Hypothetical Find Routine	120
Address Space Mapping	121
Address Resolution	123
Dump Directory Handling	125
Initialization and Termination	126
Accessing the Dump Directory	126
Dump Directory VSAM Request Routines	127
Dump Directory Record Management Routines	127
Dump Directory Records	127
Properties of the Dump Directory	128
Dump Data Formatting	129
Interface and Control Flow Routines	129
Data Type Formatting Routines	131
 Section 3. Program Organization	133
Non-Executable Modules	133
The Modifiable Module	133
Read-Only Modules	134
Initialization and Termination	136
IPCS STAE Exits	142
Monitor	143
Automatic Storage Management	144
Subcommand Monitor	145
Program Management	152
Program BLDL Table Management	152
Non-Resident Module Management	153
Subcommand Processors	155
General Subcommand Processing Modules	156
Parse Validity Checking for Subcommands	157
Interface to IKJPARS	157
IKJPARS Validity Check Routines	158
TSO Utility Commands	159
SYSDSCAN Command	159
IPCSDDIR Command	159
Session Control Subcommands	160
HELP Subcommand	160
SETDEF Subcommand	160
TSO Subcommand	162
COPYDUMP Subcommand	164
Problem and Data Management Subcommands	166
Analysis Subcommands	192
ASMCHECK Subcommand	192
COMCHECK Subcommand	192
ENQCHECK Subcommand	194
STATUS Subcommand	195
SUMMARY Subcommand	198
Exit Interface Services	202
Exit Support Subcommands	202
ASCBEXIT Subcommand	202
IOSCHECK Subcommand	203
TCBEXIT Subcommand	203
VERBEXIT Subcommand	204
Common Exit Services	205
Exit Control Table (ECT) Service	208
Select ASID Service	209
Format Service	210
Get Symbol Service	211
Equate Symbol Service	212
Line-Oriented Dump Viewing Subcommands	213
FIND Subcommand	213
Value Resolution Service	214

FINDMOD Subcommand	217
FINDUCB Subcommand	218
LIST Subcommand	218
LISTUCB Subcommand	219
RUNCHAIN Subcommand	219
Full-screen Dump Viewing (DSPL3270)	220
Common Services for CLIST Support	222
CLIST Support Subcommands	222
COMPARE Subcommand	222
EVALDEF Subcommand	224
EVALDUMP Subcommand	224
EVALMAP Subcommand	225
EVALSYM Subcommand	226
EVALUATE Subcommand	227
INTEGER Subcommand	228
ISPEXEC Subcommand	228
NOTE Subcommand	229
Dump Directory Handling Subcommands	230
OPEN Subcommand	230
CLOSE Subcommand	230
LISTDUMP and DROPDUMP Subcommands	231
DROPMAP, LISTMAP, and SCAN Subcommands	232
DROPSYM and LISTSYM Subcommands	234
EQUATE and STACK Subcommand	236
RENUM Subcommand	238
ISPF Dialog Programs	239
Logical Screen Task Setup	239
IPCS Subcommand Processing Dialog	240
Dump Display Reporter - NTRCEPT	240
ISPF Dump Viewing Dialog	244
Common Service Functions	249
Data Access Services	249
Print and Terminal Services	265
Print Routines	265
Message Routing Routines	267
Terminal Transmission Routines	271
Message Text Build and Routing	272
Storage Map Management	273
Scan Routines	280
Dump Access	282
The Buffer Manager	282
Block Read Routines	282
Storage Access Routines	283
PRDMP Exit Services	283
Dump Initialization	285
Symbol Management	294
Special Symbol Processing	296
Primary Subroutines	296
Suffix Editing Subroutines	296
Finding Data Areas	297
Address Space Mapping	299
Address Resolution	304
Dump Directory Handling	310
Simple Request Handling	310
Complex Request Handling	310
Dump Directory Record Management	313
Dump Data Formatting	315
Basic Formatters	315
Storage Formatters	315
Support Modules	316
Inter-Task Request Servers	324
Section 4. Data Areas	325
BLSABDPL	326
BLSBVT	333
BLSDMCB	336
BLSDSD	342
BLSDVT	343
BLSFP	345

BLSLBS 346
BLSLNTRC 352
BLSPDR 357
BLSQEXTP 360
BLSQSMPL 361
BLSQSRA 363
BLSRDATC 365
BLSRDATS 367
BLSRDATT 369
BLSRDTDT 370
BLSRDUT 371
BLSRESSY 373
BLSRFD 376
BLSRHSD 378
BLSRPHSY 381
BLSRPRX 382
BLSRRASY 385
BLSRRBSY 387
BLSRRCSY 388
BLSRRDSY 389
BLSRSASY 391
BLSRSDSY 394
BLSRSST 395
BLSRVALY 396
BLSRVT 397
BLSRZZ6 400
BLSUVSFB 403
BLSUZZ1 404
BLSUZZ2 406

Section 5. Diagnostic Aids 417

IPCS Return Code Table 417
IPCS User Completion (ABEND) Codes 424
 Problem Determination Table 432
 Table I 432
 ABEND Cross Reference Table 434
 IPCS ABEND Codes (by Module) 435
Error Recovery Hints 439
 ISPF Full Screen Dialog 439

Appendix A. Control Block Acronyms 441

Acronyms for MVS/XA Control Blocks Used by IPCS 441
Acronyms for IPCS Control Blocks 442
General Acronyms for IPCS Control Blocks 443

Appendix B. Common Module Characteristics 445

IPCS Register Usage 445
 Program Linkage 445
 Internal Usage 445
Patch Labels 446

Appendix C. IPCS Data Set Contention Protocols 447

Allocating Resources 447
Inter-Address Space Resource Contention 447
 High-Contention Protocol 448
 Low-Contention Protocol 449

Index 451

Figures

1. IPCS Command Task Structure	2
2. Control Block and Data Set Overview	5
3. IPCS Functional Components	7
4. IPCS Command (Processing Overview)	15
5. Monitor	19
6. IPCS Subcommand Table	22
7. IPCS Storage Management	24
8. COPYDUMP Subcommand	31
9. ADDPROB Subcommand	33
10. DELPROB Subcommand	35
11. LISTPROB Subcommand	38
12. MODPROB Subcommand	40
13. ADDDSN Subcommand	43
14. DELDSN Subcommand	47
15. LISTDSN Subcommand	51
16. MODDSN Subcommand	53
17. ASMCHECK Subcommand	56
18. COMCHECK Subcommand	58
19. ENQCHECK Subcommand	60
20. STATUS Subcommand	62
21. Simulated PRDMP Parameter List	65
22. DSPL3270 Subcommand (Overview)	72
23. DSPL3270 Subcommand (Initialization)	73
24. DSPL3270 Subcommand (Screen Generation)	74
25. DSPL3270 Subcommand (Response Processing)	75
26. DSPL3270 Subcommand (Termination)	76
27. Dialog Overview: Relationship of Subsystems	84
28. Data Access Services Overview	88
29. Data Access Services Allocation	91
30. Data Access Services Access Request	94
31. Data Access Services Deallocation	97
32. Print and Terminal Services	98
33. Service Code Constants and Their Related Services	101
34. Dump Access	108
35. Summary of Dump Tables	110
36. Dump Open Routine Summary	111
37. Storage Map Management	113
38. Symbol Management	117
39. Address Space Mapping	122
40. Address Resolution	124
41. Dump Directory Handling	126
42. Data Types and Output Formats	129
43. Dump Data Formatting	130
44. Initialization and Termination Control Flow	136
45. Subcommand Monitor Function Control Flow	145
46. Program Management Control Flow	152
47. IPCSDDIR Command Control Flow	159
48. SETDEF Subcommand Control Flow	161
49. COPYDUMP Subcommand Control Flow	164
50. ADDPROB Subcommand Control Flow	166
51. DELPROB Subcommand Control Flow	167
52. LISTPROB Subcommand Control Flow	168
53. MODPROB Subcommand Control Flow	168
54. ADDDSN Subcommand Control Flow	169
55. DELDSN Subcommand Control Flow	170
56. LISTDSN Subcommand Control Flow	171
57. MODDSN Subcommand Control Flow	172
58. ASMCHECK Subcommand Control Flow	192
59. COMCHECK Subcommand Control Flow	193
60. ENQCHECK Subcommand Control Flow	194
61. STATUS Subcommand Control Flow	196
62. SUMMARY Subcommand Control Flow	198
63. ASCBEXIT Subcommand Control Flow	202
64. IOSCHECK Subcommand Control Flow	203
65. TCBEXIT Subcommand Control Flow	204
66. VERBEXIT Subcommand Control Flow	204

67. Common Exit Services Control Flow	205
68. ECT Services Control Flow	208
69. Select ASID Service Control Flow	209
70. Format Service Control Flow	210
71. Get Symbol Service Control Flow	211
72. Equate Symbol Service Control Flow	212
73. FIND Subcommand Control Flow	213
74. FINDMOD Subcommand Control Flow	217
75. FINDUCB Subcommand Control Flow	218
76. LIST Subcommand Control Flow	218
77. LISTUCB Subcommand Control Flow	219
78. RUNCHAIN Subcommand Control Flow	220
79. DSPL3270 Subcommand Control Flow	221
80. COMPARE Subcommand Control Flow	223
81. EVALDEF Subcommand Control Flow	224
82. EVALDUMP Subcommand Control Flow	225
83. EVALMAP Subcommand Control Flow	226
84. EVALSYM Subcommand Control Flow	227
85. EVALUATE Subcommand Control Flow	227
86. INTEGER Subcommand Control Flow	228
87. ISPEXEC Processing Control Flow	229
88. NOTE Subcommand Control Flow	229
89. OPEN Subcommand Control Flow	230
90. CLOSE Subcommand Control Flow	230
91. Summary of LISTDUMP and DROPDUMP	231
92. DROPMAP Subcommand Control Flow	232
93. LISTMAP Subcommand Control Flow	232
94. SCAN Subcommand Control Flow	233
95. DROPSYM Subcommand Control Flow	234
96. LISTSYM Subcommand Control Flow	235
97. EQUATE Subcommand Control Flow	236
98. STACK Subcommand Control Flow	237
99. RENUM Subcommand Control Flow	238
100. ISPF/IPCS Environment Control Flow	239
101. Subcommand Processing Dialog Control Flow	240
102. Dump Display Reporter Control Flow	241
103. ISPF Dump Viewing Dialog Control Flow	245
104. Allocation Control Flow	249
105. Deallocation Control Flow	249
106. Data Access Request Control Flow	250
107. Print Transmission Routine Summary	265
108. Message Routing Routine Summary	268
109. Terminal Transmission Routine Summary	271
110. Storage Map Management Control Flow	274
111. Scan Routines Control Flow	280
112. Summary of Scan Routines	281
113. Summary of Storage Access Routines	283
114. Summary of PRDMP Exit Storage Access Routines	284
115. Dump Initialization Control Flow	285
116. Symbol Retrieval Routine Summary	295
117. Symbol Table Update Routine Summary	295
118. Finding Data Areas Control Flow	297
119. Summary of Find Routines	298
120. Address Space Mapping Control Flow	300
121. Address Resolution Control Flow	304
122. Dump Directory Simple Request Handling Summary	310
123. Dump Directory Complex Request Control Flow	311
124. Dump Directory Record Management Summary	314
125. Summary of Basic Formatters	315
126. Summary of Storage Formatters	316
127. Inter-Task Service Summary	324
128. Standard Return Code Summary	417

Summary of Amendments

Summary of Amendments for LY28-1298-2 MVS/System Product Version 2 Release 1.7

This major revision consists of changes to support MVS/System Product Version 2 Release 1.7. The changes include:

- For the Vector Facility Enhancement, additions to the STATUS subcommand, the scan routines, and the data area, BLSABDPL.
- Changes to the data area, BLSRRASY

Summary of Amendments for LY28-1298-1 MVS/System Product Version 2 Release 1.2

This major revision incorporates technical and editorial updates in support of MVS/System Product Version 2 Release 1.2. The changes include:

- The addition of two new IPCS subcommands, STACK and RENUM.
- The addition of the EXEC keyword on the RUNCHAIN subcommand
- The enhancement of the BROWSE dialog program
- The ability to view dump data in full screen mode using the dump display reporter facility
- The addition of the FILE and DDNAME keywords, which permit dumps that are stored on tape to be accessed
- The addition of the ACTIVE, MAIN, and STORAGE keywords, which permit the accessing of the address space in which IPCS is currently executing
- The enhancement of the dump title that IPCS provides
- The enhancement of the SUMMARY subcommand to support new keywords that provide a variety of output formats and provide a selectivity of address spaces

- The addition of the following new common exit services:
 - Exit services router
 - Select ASID
 - Get symbol
 - Equate symbol
 - Control block formatter
 - Format model processor
 - Exit control table (ECT)
- Minor technical and editorial changes throughout the publication, including some reorganization of information.

Section 1. Introduction

This section contains general information about the purpose of the interactive problem control system (IPCS), its relationship to the operating system, its task and data structures, and its functional components.

Purpose

MVS/XA IPCS provides an interactive mechanism for the online viewing and analyzing of machine readable dumps, and provides functions to assist in the management of system software problems and their associated data. A list of these functions appears under “IPCS Subcommand Processors” on page 8.

Relationship to the Operating System

IPCS runs as a TSO command processor that is invoked by issuing the IPCS command. IPCS receives control from the calling program (TSO terminal monitor program (TMP) or another program that supplies the command processor parameter list (CPPL) ATTACH interface). Once IPCS receives control, it establishes an environment in which the user communicates by invoking IPCS subcommands. When the user indicates that processing is complete, IPCS returns control to the calling program.

IPCS communicates with MVS/XA and TSO for input/output and other services. A general description of the control program and its services is given in the publications:

- *TSO Guide to Writing a TMP or a Command Processor*
- *Data Management Services Guide*
- *System Macros and Facilities*

Task Structure

As an attached command processor under TSO, IPCS is responsible for establishing an environment and control structure that permit and support the processing of subcommands, absorb a minimum of MVS/XA system resources, and protect the integrity of IPCS and system resources. The task structure that is established is shown in Figure 1 on page 2.

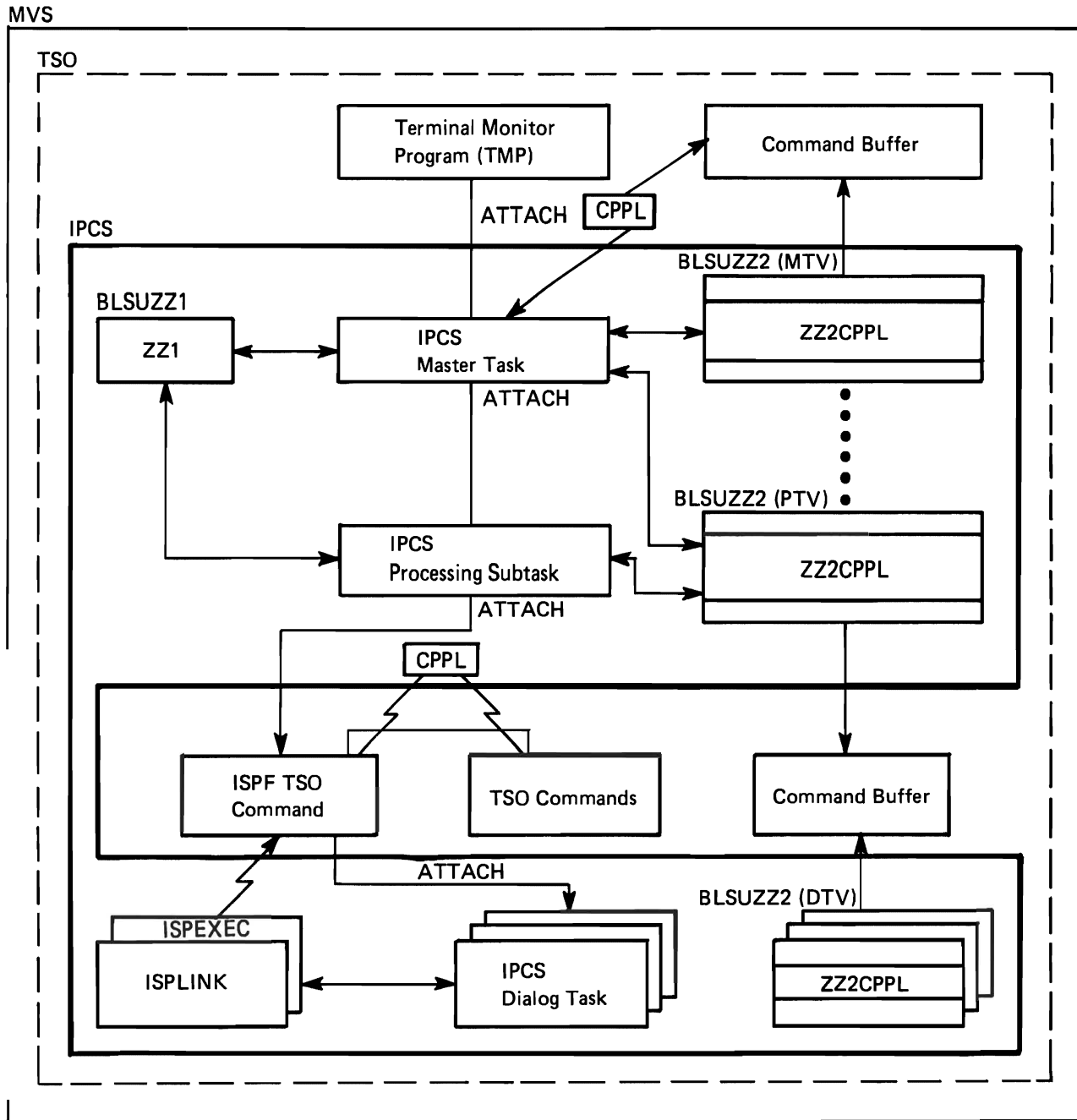


Figure 1. IPCS Command Task Structure

Master Task

The master task functions as an anchor for IPCS processing. It serves a largely passive role during most of an IPCS session, leaving the majority of IPCS processing responsibilities to an attached subtask. The following functions are accomplished under the master task:

- IPCS initialization and termination that includes the creation of a processing subtask
- Intertask communication

Intertask (as well as intermodule) communication relies heavily on the existence of two major IPCS control blocks:

- The common area (BLSUZZ1) contains information shared by all tasks in the IPCS environment. Early in IPCS' initialization, the master task establishes BLSUZZ1.
- The task variable (BLSUZZ2) is an n-element chain of task-related control blocks. The elements of the chain are associated either with the master task, the processing subtask, or a dialog task and contain information pertinent to the specific task:
 - The master task variable (MTV) is the first element in the BLSUZZ2 chain and contains information associated with the IPCS master task. The MTV is created early in IPCS initialization processing.
 - The processing task variable (PTV) is the second element in the BLSUZZ2 chain and contains information associated with the IPCS processing subtask. Before the master task variable (MTV) attaches the IPCS processing subtask, the MTV creates the PTV.
 - Dialog task variables (DTVs) occupy the third and subsequent positions on the chain and contain information associated with an ISPF logical screen task that has been prepared by dialog program BLSG for the use of IPCS services.
 - The active task variable (ATV) is a generic name used throughout this document to refer to the element in the BLSUZZ2 chain on whose behalf a service is being requested or performed.

Processing Subtask

The processing subtask is attached by the master task as a final step in initialization processing. The processing required to recognize and respond to IPCS subcommand invocations is performed under this subtask. Non-IPCS subcommands (for example, TSO commands) are executed under a separately attached subtask created when the processing subtask identifies the command verb.

ISPF Logical Screen Task

An interactive system productivity facility (ISPF) TSO command attaches an ISPF logical screen task. The dialog program, BLSG, prepares that ISPF logical screen task for IPCS' use. While BLSG is active for the task, dialog programs BLSGSCMD and BLSLDISP might be used to process IPCS subcommands,

including those that request ISPF services. Unless otherwise stated, references to the processing subtask throughout this publication are applicable to an ISPF logical screen task as well.

Data Structure

The major control blocks and direct access data sets established and used by IPCS are shown in Figure 2 on page 5 and summarized in the following discussion.

Control Blocks

Block	Description
BLSUZZ1	The common area contains information shared by all tasks in the IPCS environment.
BLSUZZ2	The task variable is an n-element chain of task-related control blocks, each of which contains IPCS data used by one MVS task.
BLSUVSFB	BLSUVSFB contains the access control block (ACB) and related information associated with the processing of the dump directory.
BLSRZZ6	BLSRZZ6 contains the DCB and related information associated with the processing of the dump data set.
BLSRDUT	The dump table identifies the special processing required for each type of dump data set.
BLSRDTDT	A data type processing table identifies AREAs and STRUCTUREs for which special support is provided for each type of dump data set.
BLSRSST	A special symbol table identifies special symbols supported for each type of dump data set.
BLSDMCB	A data access services control block contains the information associated with a data set processed by the data access services function. Three such data sets are: • SYS1.PARMLIB session parameters member • The data set directory • The problem directory
BLSFP	The facility parameter block contains the IPCS session control parameters and is constructed from the contents of the IPCS SYS1.PARMLIB member.

Direct Access Data Sets

The direct access data sets that IPCS uses are:

Problem Directory: The problem directory is used to record information about the problems that have been reported to IPCS.

Data Set Directory: The data set directory is used to record information about the data sets that have been associated with problems.

Dump Directory: The dump directory is a VSAM data set used by IPCS to accumulate information describing the contents and physical characteristics of dump data sets.

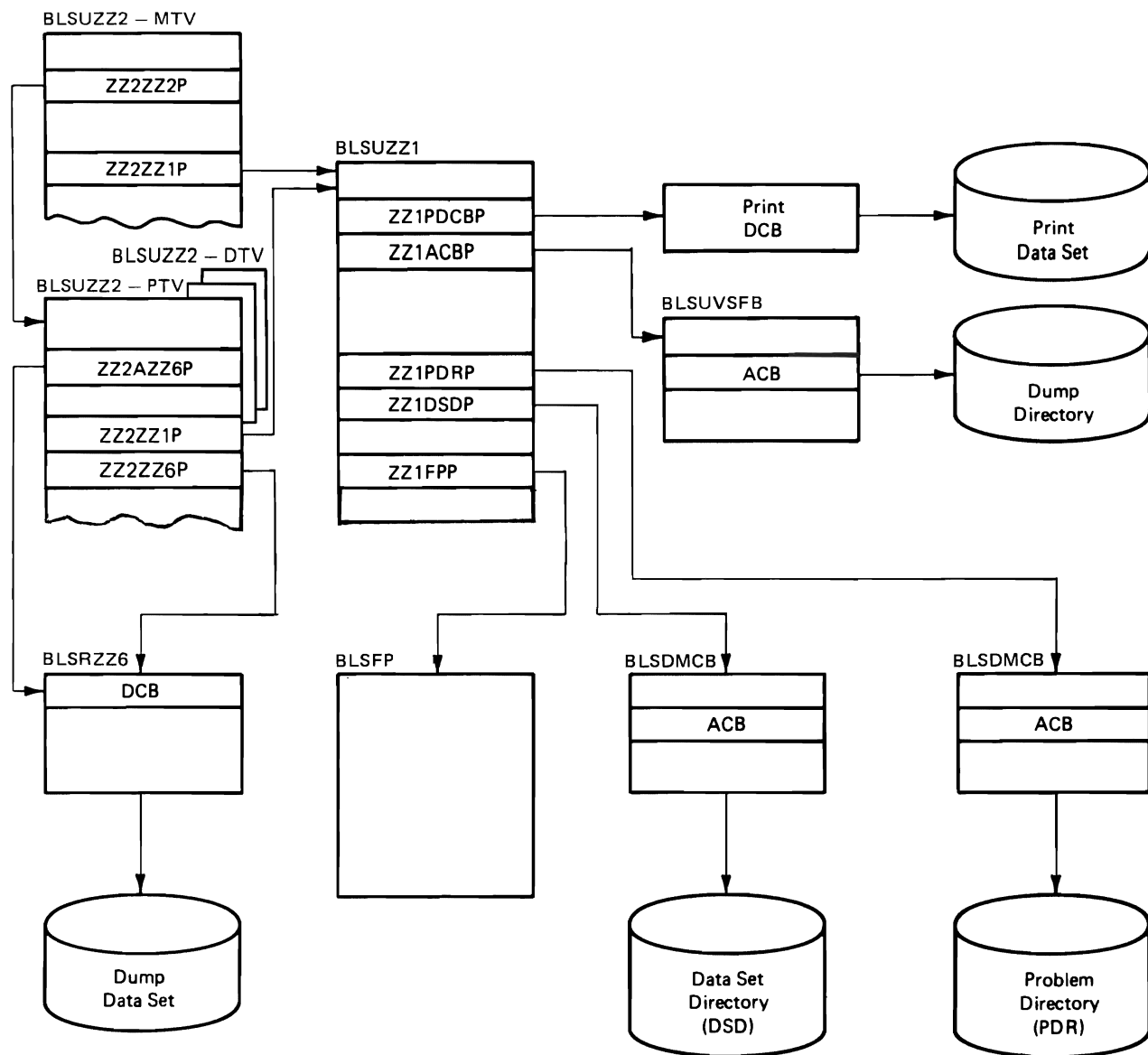


Figure 2. Control Block and Data Set Overview

Functional Components

IPCS consists of five major functional components:

- Initialization and termination
- Monitor
- IPCS subcommand processors
- ISPF dialog programs
- Common services

These components are shown in Figure 3 on page 7.

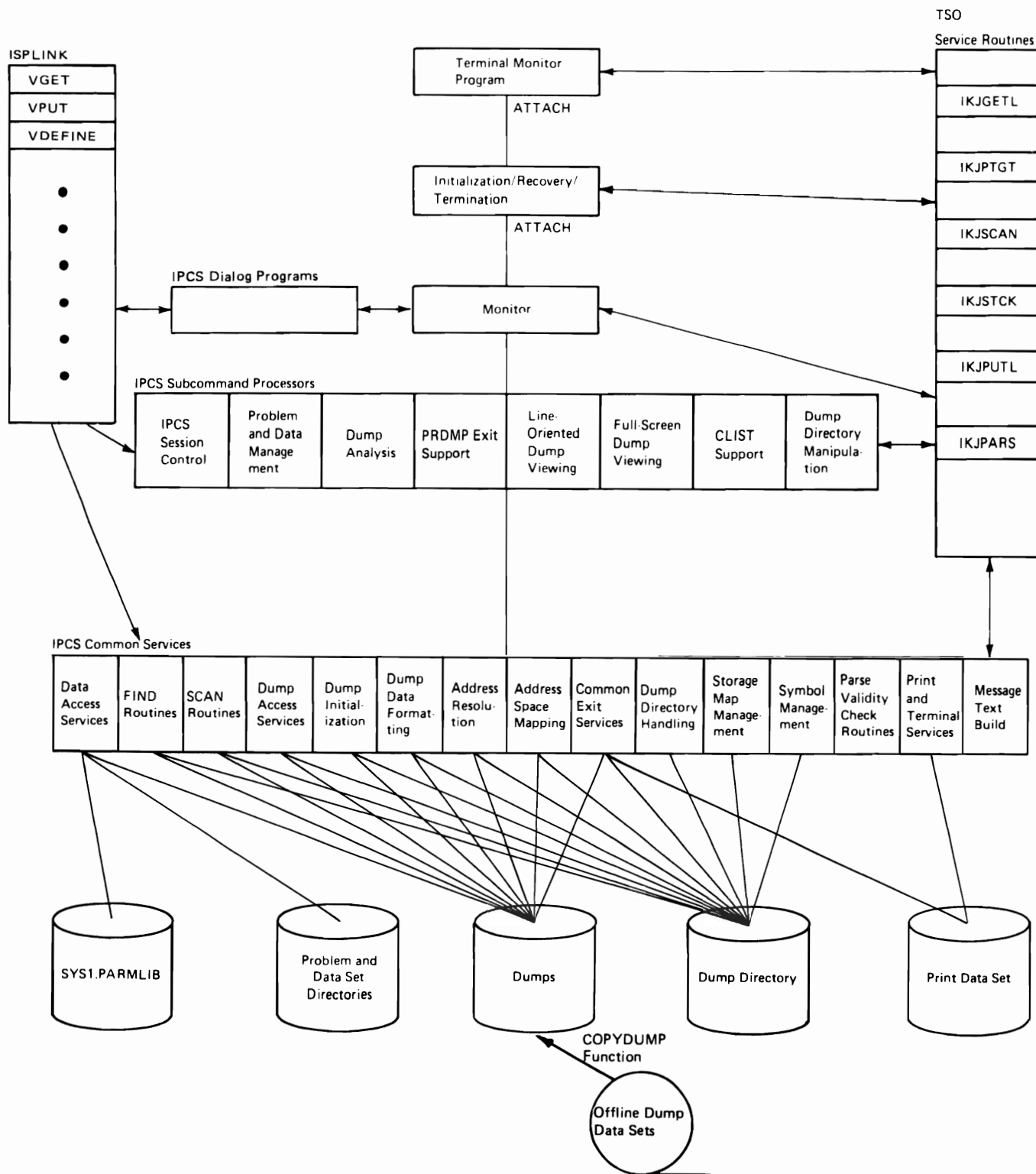


Figure 3. IPCS Functional Components

An overview of the processing performed by these components follows.

Initialization and Termination

After receiving control from the TSO TMP, the IPCS master task (load module BLSU) loads BLSUPUT. BLSUPUT contains the skeletons for the major control blocks for IPCS execution. BLSU performs a partial initialization of the IPCS common area (BLSUZZ1), which is a control block skeleton in BLSUPUT. BLSU then links to load module BLSUINI1 to provide IPCS session initialization. BLSUINI1 establishes the necessary environment for IPCS subcommand execution.

Once the environment is established, BLSU passes control to BLSUMOAT. BLSUMOAT attaches the processing subtask (BLSUINI2) with the ESTAI option. After attaching the processing subtask, the BLSUMOAT remains dormant until the processing subtask:

- Terminates normally at the end of the IPCS session
- Requests cross-task communication
- Terminates abnormally

Monitor

After receiving control from the master task, the processing subtask (BLSUINI2) completes the initialization of the processing task variable and establishes the necessary subtask environment (that is, the monitor) for IPCS subcommand execution. Three related functions are performed under the monitor:

- *Subcommand retrieval and execution.* Module, IKJPTCT, retrieves subcommands entered by the user. The subcommand verb is isolated and associated with an IPCS subcommand processor or with a TSO command processor. Then control is given to the appropriate IPCS or TSO processor.
- *Storage management.* A collection of monitor routines that are invoked during module entry and exit processing control the use of automatic storage by IPCS routines.
- *Program management.* The program management routines manage the transfer of control between nonresident IPCS routines. Tables are maintained that allow program management to determine if an invoked routine is resident. If so, control is passed to the designated routine. Otherwise, the routine is loaded and the program management tables are updated with the routine's address before the transfer of control.

IPCS Subcommand Processors

Each IPCS subcommand processor executes in two phases:

1. Uses IKJPARS to obtain a tabular description of subcommand operands.
2. Performs the operations specified by the subcommand.

The IPCS subcommands can be divided into eight logical groups:

- IPCS session control (including copying dumps)
- Problem and data management
- Dump analysis
- PRDMP exit support

- Line-oriented dump viewing
- Full-screen dump viewing
- CLIST support
- Dump directory manipulation

ISPF Dialog Programs

IPCS supplies three ISPF dialog programs that can be used when ISPF is (directly or indirectly) attached by the IPCS command:

BLSG	prepares an ISPF dialog task for the use of the IPCS services provided by BLSGSCMD and BLSLDISP.
BLSGSCMD	processes an IPCS subcommand or a CLIST composed of IPCS subcommands and CLIST control statements.
BLSLDISP	provides support for full-screen examination of dump data.

Both BLSGSCMD and BLSLDISP allow IPCS subcommands and CLISTs to be requested using ISPF panels. The following IPCS functions are provided only by subcommands invoked in this manner:

- The ISPEXEC subcommand (BLSGX) can be used. This subcommand provides access to a wide variety of services supported by ISPF for dialogs.
- The DIALOG keyword can be used on ADDPROB, EVALDEF, EVALDUMP, EVALMAP, EVALSYM, EVALUATE, and INTEGER subcommands. This keyword instructs IPCS subcommands to format information into ISPF variables where it can be used by ISPF.

IPCS Common Service Functions

Although the service modules of the common service functions are distributed among the load module structure of IPCS, the common services can be classified into general categories as shown in the following list:

Data Access Services
Print and Terminal Services
Message Text Build and Routing
Common Exit Services
Dump Access
Dump Initialization
Storage Map Management
Scan Routines
Symbol Management
Find Routines
Address Space Mapping
Address Resolution
Dump Directory Handling
Dump Data Formatting

These common service functions are used by the IPCS subcommands. A brief introduction to each of the categories follows.

Data Access Services

The data access services function provides direct access to the problem directory and data set directory. This service implements resource serialization conventions

to ensure the data integrity of these shared resources. Data access services process requests to allocate, open, read, write, close, and free resources.

Print and Terminal Services

Print and terminal service routines simplify the transmission of messages:

- Message routing service routines (BLSUPUTx) permit the same message text to be routed to the TSO output medium, the IPCS print data set (IPCSPRNT), or both. Appropriate message editing is performed to ensure that messages are compatible with the format requirements of the output destinations.
- Print file management services (BLSUPRTx) transmit messages to the print data set. They also ensure that the page depth and page width prescribed by the user are observed.
- Terminal management services (BLSUTRMx) prepare messages for processing by the TSO PUTLINE service and transmit the messages via that service.

Message Text Build and Routing

The message text build and routing routine (module BLSDMSG0) dynamically builds messages for IPCS problem management, data management, and data access services. The message is built from a message skeleton contained in a message CSECT (BLSDMSGs) and a list of inserts. Functions provided include terminal blank compression for inserts, internal message chain construction and maintenance, and scope of message routing. BLSDMSG0 uses the print and terminal services to transmit message chains when needed.

Common Exit Services

Common exit services routines (BLSQxxxx) provide dump processing exits with the capability of controlling the displayed or printed dump data output. The services available through the exit services router service are:

- Storage access
- Print
- Format model processor
- Control block formatter
- Index
- Exit control table (ECT)
- Get symbol
- Equate symbol
- Select ASID

When an IPCS, PRDMP, or SNAP exit requests a common exit service, the exit services router, BLSQROUT, provides the common interface. BLSQROUT invokes all services using BLSABDPL, the exit parameter list.

Dump Access

Dump access service routines (BLSRACCx) handle all direct accesses to dump data sets. The routines use a pool of dump record buffers to satisfy a significant portion of requests from virtual storage buffers rather than requesting the transfer of a block from a dump data set to virtual storage. Buffer management data is maintained in a compact area, separate from the buffer pool, to minimize page faults.

Dump Initialization

Dump initialization service routines (BLSRDUxx) determine whether the information required to process the current IPCS data set as a dump is available when it is required. If the information is not available, it is established by dump initialization.

A dump initialization service routine, BLSRDUDR, is provided to erase the description of a dump data set from the directory.

Storage Map Management

Storage map management service routines (BLSRSAxx) manipulate dump directory records that:

- Enumerate blocks of storage that have been identified to IPCS.
- Describe the results of validation performed by IPCS for those blocks for which special support has been provided.

Scan Routines

The scan routines (BLSVxxxx) check the validity of data areas found in a dump. There is one routine for each type of data area for which validation is performed. Each scan routine centralizes the tests done by IPCS for a particular data area. The results of each validation are saved in the storage map; IPCS does not need to validate any area more than once.

Symbol Management

Symbol management service routines (BLSRESxx) manipulate dump directory records that enumerate symbols that have been associated with storage in a dumped system. A central service routine, BLSRESGE, determines when it is appropriate to isolate data.

Find Routines

Find routines (BLSSxxxx) locate particular data areas on demand. There is a different find routine for each type of data area supported. Each routine establishes a naming convention that uniquely identifies each instance of its data area type. The results of searches are saved in the symbol table.

Address Space Mapping

Address space mapping service routines (BLSRRAxx) manipulate dump directory records that associate storage from an address space in a dumped system with records contained in a dump data set. A central service routine, BLSRRAGE, manages the process of address space mapping.

Address Resolution

Address resolution service routines (BLSRADDx) process the data description operands specified on IPCS subcommands, changing PDEs produced by IKJPARS into structures used by IPCS modules to describe the properties of storage.

Dump Directory Handling

Dump directory handling routines (BLSUVSxx) request VSAM services related to the dump directory (IPCSDDIR) and diagnose errors detected by VSAM during that processing. Dump directory record management routines (BLSRrrff where

“rr” indicates the type of record and “ff” indicates the function performed) perform common storage and retrieval operations for specific types of dump directory records.

Dump Data Formatting

Dump data formatting service routines (BLSTxxxx) are used to display dumped storage. Available data plus user preferences expressed via subcommand keywords control the amount of information displayed.

Section 2. Method of Operation

This section follows the five major functions introduced in Section 1:

- The IPCS command (including initialization, task communication, and termination)
- Monitor
- Subcommand processors
- ISPF dialog programs
- Common service routines

In addition, the section begins with a description of the two TSO utility commands that help set up the dump processing environment.

TSO Utility Commands

There are two TSO utility commands supplied with IPCS, in addition to the IPCS command. The first command, SYSDSCAN, displays titles from SYS1.DUMP data sets. The second, IPCSDDIR, initializes the dump directory to be used by IPCS subcommands.

The execution of these commands is independent of IPCS processing. No IPCS common service functions are used by these commands. The commands receive the standard TSO command processor parameter list (CPPL), use IKJPARS to parse the command, and use IKJPUTL to issue messages. For more information, consult the Command Processor section of the *TSO Guide to Writing a TMP or a Command Processor*.

SYSDSCAN Command

SYSDSCAN dynamically allocates each of a given range of SYS1.DUMP data sets, opens them, reads the header record, and displays the title of the dump. BLSRDUSC is the module that processes the SYSDSCAN command.

First, IKJPARS is called to parse the operands. If a nonzero return code is returned, an exit is made with that return code. The range entered or implied is used as the last two digits (nn) of the name SYS1.DUMPnn for each dump examined. Validity-check exit BLSUVP99 invoked during the parse ensures a range of 00-99 and proper order.

Each number in the range is converted to character and appended to the characters "SYS1.DUMP." SVC 99 (dynamic allocation) is invoked. If a nonzero return code results, module IKJEFF18 informs the user of the situation, and processing continues with the next number in the range.

If the data set was allocated correctly, its DSORG is examined and the user informed if it is inconsistent with that of a dump data set. If OPEN fails, the user is informed. (An OPEN at this point should not fail unless another task has deleted the data set after this task has allocated it.)

If the OPEN is successful, up to 10 records are read searching for the header record. If found, the date, time, and title are displayed (all displays are via IKJPUTL). If no header record is found, or if the data set is empty, the user is notified. Each data set is dynamically deallocated after use.

IPCSDDIR Command

IPCSDDIR initializes an IPCS dump directory. The directory is a key-sequenced VSAM data set. (See “Dump Directory Handling” later in this section.) IPCSDDIR writes beginning and ending records in the directory. One record has a key of all X'00' and the other has a key of all X'FF'. BLSRVSLD is the module that processes the IPCSDDIR command.

BLSRVSLD parses (IKJPARS) the command line to obtain the name of the dump directory data set. BLSRVVPS is used to partially validity check the data set name.

BLSRVVPS validity checks an IKJPOSIT PDE that contains the data set name of a dump directory which requires initialization. The validity check fails if a member name is present.

The named data set is dynamically allocated. If any errors occur, DAIRFAIL (IKJEFF18) issues diagnostics and the command is terminated. An ACB and an RPL are generated, and the directory is opened. Errors during dump directory processing are diagnosed by VSAMFAIL (IKJEFF19).

The low and high key records are written (PUT macro). Any errors that occur are diagnosed but ignored. An attempt is always made to write both records, even if one write fails.

The directory is then closed and deallocated, and control is returned to the caller.

IPCS Command

IPCS is a command processor which receives control as the result of an ATTACH of load module BLSU (alias IPCS). BLSU is the entry point of and the controlling module within load module BLSU. Most processing associated with the IPCS command is performed by the monitor function, the subcommand processors, and the common service modules, each of which is described later. BLSU and the routines it invokes satisfy the following global requirements of the IPCS command processor:

- Initialize the environment expected by the IPCS subcommand processors. Included in initialization are such functions as allocating and opening the IPCS data sets, establishing system parameters, fetching load modules, constructing requisite control blocks, and attaching the processing subtask.
- Support a limited intertask communication capability. Intertask communication within IPCS occurs as a part of the processing of dump data sets and eases recovery and cleanup during abnormal termination processing.
- Terminate the IPCS environment. Termination processing ensures that all resources used during IPCS processing are returned to MVS. Included in this step are the closing and deallocation of IPCS data sets, freeing of GETMAINed storage, and deletion of fetched load modules.

See Figure 4 for an overview of the processing that results from the IPCS command.

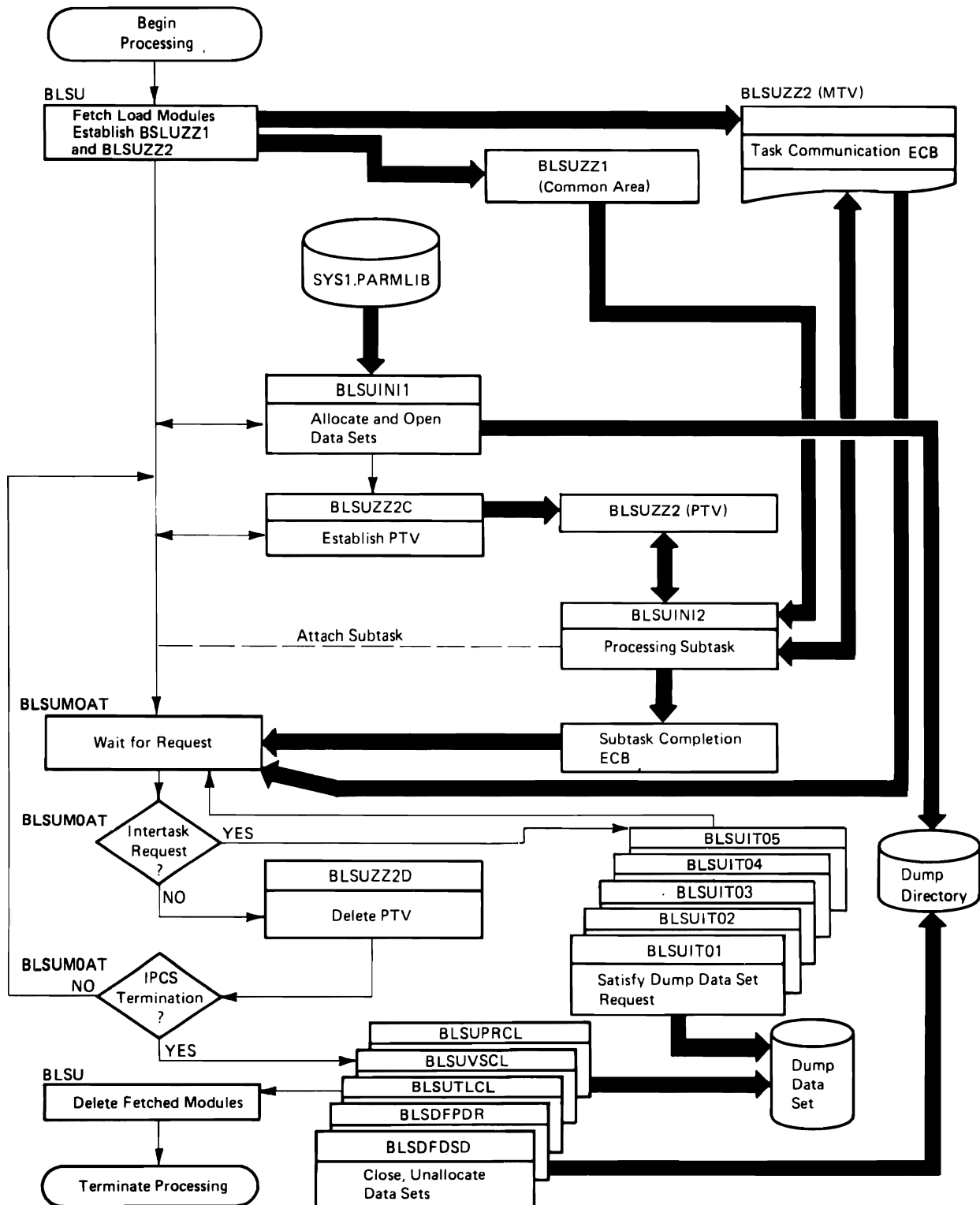


Figure 4. IPCS Command (Processing Overview)

Initialization

The initialization portion of BLSU functions as a bootstrap operation in that its first concern is to dynamically fetch the minimum subset of data areas (BLSUZZ1 and BLSUZZ2) and load modules essential to the continuation of initialization. Early in its processing, BLSU establishes the master task ESTAE routine (BLSUSTAE) and initializes key fields in control blocks BLSUZZ1 and BLSUZZ2 (BLSDINI0 and BLSUINI0). Having accomplished this, BLSU links to BLSUINI1, where much of the IPCS initialization processing is performed.

BLSUINI1 performs the following functions:

- The operands associated with the IPCS command are parsed (IKJPARS). If parsing is not successful, IPCS initialization is halted.
- SYS1.PARMLIB member IPCSPRnn is processed (BLSDIFP0). After allocating SYS1.PARMLIB (BLSUDYNA), BLSDIFP0 LINKs to BLSDIFP1 to process the session parameters member.

BLSDIFP1 employs the DATASET option of the TSO STACK macro to open and read the contents of member IPCSPRnn. This facilitates the use of TSO service routines IKJGETL and IKJPARS during the validation of the PARMLIB information. As a part of validation, the IPCS problem directory (PDR) and data set directory (DSD) data sets are allocated. If PARMLIB validation is successful, the IPCS facility parameters control block (FPBLOK), which defines the operative system parameters for the current IPCS session, is established. Unsuccessful PARMLIB processing causes initialization to be halted. SYS1.PARMLIB(IPCSPRnn) is closed and deallocated at the completion of session parameters member processing.

- The dump directory data set (IPCSDDIR) is opened (BLSUVSOP and BLSUVSOQ). BLSUVSOP determines if IPCSDDIR references a VSAM data set. BLSUVSOQ completes preparations to use the dump directory:
 - It determines if IPCSDDIR references a VSAM CLUSTER (SHOWCAT macro) whose characteristics are consistent with those required for an IPCS dump directory.
 - It allocates a local shared resource pool containing buffers matched to the control interval sizes of the index and data portions of the dump directory (BLDVSRP macro).
 - It opens the dump directory for update processing (OPEN macro).
 - It determines whether the dump directory can be used (VERIFY macro) when OPEN return code 4 indicates potential problems.

IPCS initialization is halted if the dump directory data set has incorrect characteristics or cannot be prepared for use.

- The list of TASKLIB data sets is allocated, concatenated in the order specified, and a DCB is opened to the resulting concatenation (BLSUTLOP). As a part of TASKLIB allocation, BLSUTLOP verifies that each data set has partitioned organization. IPCS initialization is halted if a TASKLIB data set has incorrect characteristics or cannot be opened.

Following return from BLSUINI1, BLSU invokes BLSUMOAT to pass control to the IPCS processing task. BLSUMOAT invokes BLSUZZZC to create a second task variable (see "BLSUZZ2" on page 406). This task variable, known as the processing task variable (PTV), is distinct from and in addition to the master task variable (MTV). The PTV is added to the end of the BLSUZZ2 chain and the majority of its fields are initialized.

The final step in the initialization portion of BLSUMOAT is to ATTACH (specifying BLSUSTAI as the ESTAI routine) the processing subtask, passing it the address of the newly created PTV. BLSUINI2 serves as the entry point of the attached subtask and, as its first function, completes the initialization of the PTV. BLSUINI2 also establishes the processing subtask ESTAE routine, BLSDSTAE.

At this point, the environment necessary for successful IPCS execution has been established and initialization is complete. BLSUMOAT then awaits the posting of one of two ECBs, one which signals an intertask communication request, and another which signals the completion of the processing subtask.

Intertask Communication

An intertask communication ECB is defined in the IPCS MTV. This ECB is posted by the processing subtask to alert BLSUMOAT to an outstanding dump data set service request. Five service request types are recognized:

- DYNALLOC (BLSUIT01)
- OPEN SVC (BLSUIT02)
- CLOSE SVC (BLSUIT03)
- Open print file (BLSUIT04)
- Close print file (BLSUIT05)

The specific request type is indicated in the PTV associated with the requesting subtask. BLSUMOAT intertask communication processing consists of:

- Scanning the BLSUZZ2 control block chain searching for an outstanding service request.
- Invoking the appropriate service routine when an outstanding request is encountered.
- Posting an ECB in the PTV which signals the completion of request processing.
- Continuing the scan of the BLSUZZ2 chain until all outstanding requests have been satisfied.

Termination

IPCS termination processing is of two types:

- Processing subtask termination.
- Master task termination.

A description of each type of processing follows.

Processing Subtask Termination

The completion of the processing subtask is indicated by the posting of its associated ECB. The termination may have been either normal or abnormal, as indicated by the value of flags in the IPCS common area.

Normal subtask termination is equivalent to a request to terminate the IPCS command and master task termination processing is initiated.

The processing subtask ESTAE routines (BLSDSTAE and BLSRSTAE) are entered when the subtask terminates abnormally.

BLSDSTAE	Ensures the integrity of the problem and data set directory data sets.
BLSRSTAE	Closes all dump data sets in use by the task.

Control then passes to the ESTAI exit routine, BLSUSTAI, specified when the processing subtask was attached. BLSUSTAI performs limited termination processing.

Upon recognizing an abnormal subtask termination condition, BLSUMOAT deletes the current active task variable (ATV) from the BLSUZZ2 chain (BLSUZZ2D), detaches the terminating subtask, establishes a new ATV, and reattaches the processing subtask.

Master Task Termination

The goals of master task termination processing are to protect the integrity of key IPCS data sets and to restore, as completely as possible, the status of MVS/XA to that which existed prior to the invocation of IPCS. The manner and extent to which these goals are achieved are a function of the termination type (normal or abnormal) and, in the case of an abnormal termination, the system resources available for use by termination processing.

Normal master task termination processing is performed within modules BLSU and BLSUMOAT. Master task termination is the result of a normal termination of the attached processing subtask. Normal termination processing is accomplished in the following steps:

- The processing subtask is detached.
- The PTV is deleted from the BLSUZZ2 control block chain (BLSUZZ2D).
- All IPCS data sets are closed (modules BLSDFDSD, BLSDFPDR, BLSUPRCL, BLSUTLCL, and BLSUVSCL are invoked to accomplish this processing).
- The ESTAE exit established during IPCS initialization is canceled.
- Fetched load modules are deleted (BLSUINI9 and BLSU).

Processing of the IPCS command is complete at this point. Control is returned to the TSO TMP.

The master task ESTAE exit routine (BLSUSTAE), established during initialization, is entered when an unrecoverable error is encountered during IPCS processing in the master task. Only a subset of the normal termination processing steps are attempted by BLSUSTAE. If a system diagnostic work area (SDWA) is

provided to BLSUSTAE, the IPCS data sets are closed as described above. Otherwise, no processing is attempted. In either case, no retry attempt is made and the abend is allowed to continue.

Monitor

The monitor is comprised of three functions:

- Subcommand monitoring
- Storage management
- Program management

The operation of the monitor is shown in Figure 5.

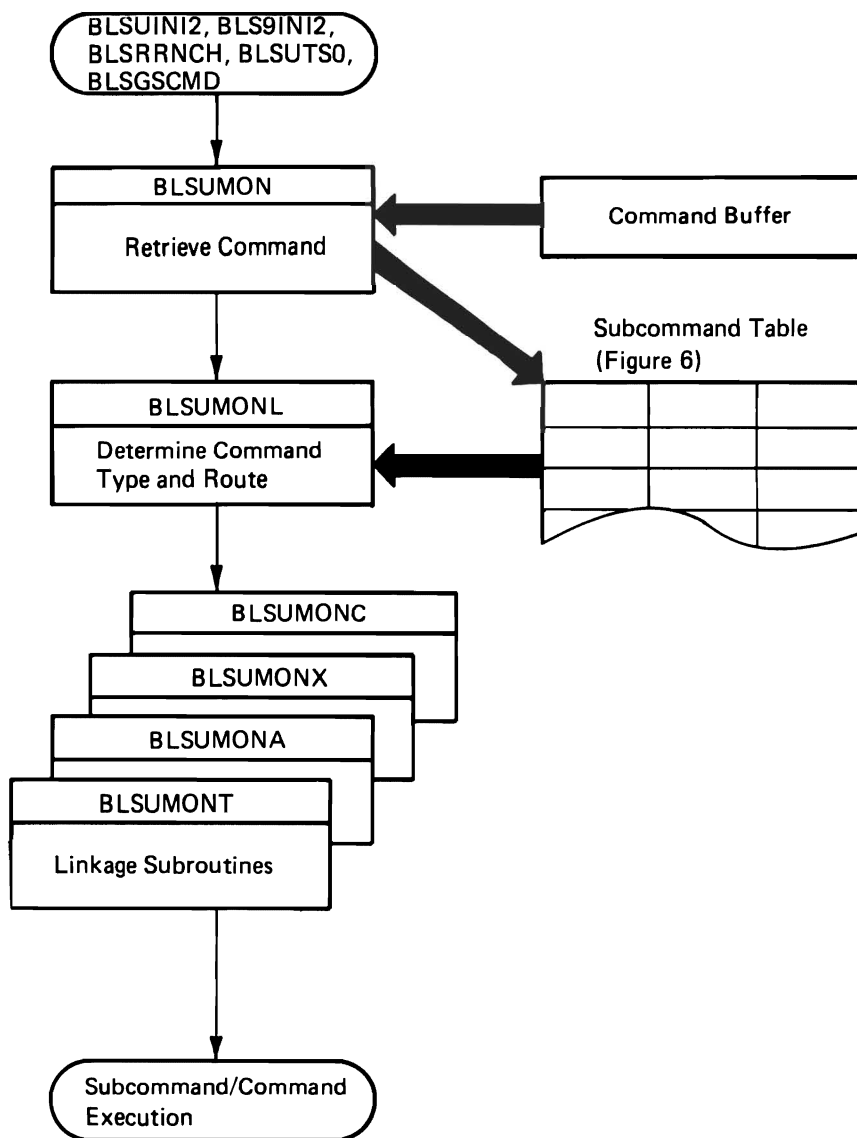


Figure 5. Monitor

Subcommand Monitoring

The monitor serves as the central point from which all command, subcommand, and CLIST processing is initiated. Its purpose is to receive each subcommand submitted during the course of an IPCS session, associate the verb specified with an entry point designed to process any subcommand beginning with the verb, and to pass control to that entry point.

Subcommand Selection

IPCS monitor subcommand processing is entered at entry point BLSUMON for two closely related purposes:

1. To solicit subcommands and CLIST invocations via the IKJPTGT TSO service routine until an **END** subcommand is entered. BLSUINI2 invokes BLSUMON during the initialization of the IPCS TSO command. BLS9INI2 also invokes BLSUMON when the TSO subcommand is entered without any operands.

The callers (BLSUINI2 or BLS9INI2) supply an address of a subcommand table (BLUSTBL or BLS9STBL) and a mode message text that BLSUMON is to display, either "IPCS" or "TSO." In addition, the callers also use STAX to indicate to the TSO terminal I/O controller that an IPCS module (BLSUMONI or BLS9MONI) is available to process a terminal attention interrupt.

2. To process an existing list of subcommands or CLIST invocations already in storage and any subcommands or CLIST invocations added to the list during BLSUMON's processing. BLSUMON's callers are: BLSRRNCH (the RUNCHAIN subcommand), BLSUTSO (the TSO subcommand) and the BLSGSCMD dialog program. When the list is depleted, BLSUMON returns to the caller without soliciting additional input via IKJPTGT.

The callers supply an address of a subcommand table (BLUSTBL or BLS9STBL) and a null mode message text to prevent BLSUMON from going to the terminal for additional input. In addition, the callers also use STAX to indicate to the TSO terminal I/O controller that an IPCS module (BLS9MONI) is available to process a terminal attention interrupt.

When a dialog program (BLSLDISP or BLSGSCMD) invokes BLSUMON, the dialog program performs an additional function just prior to termination. BLSUMON invokes BLSLMON, the dump display reporter, to present a final view of the dump report and to release any resources accumulated during the generation of that report.

BLSUMON obtains subcommand buffers from one of two sources:

- BLSUMONI queues a subcommand buffer when a terminal attention interrupt is generated and the user enters a subcommand containing a verb other than **TIME** or **ABEND**.
- IKJPTGT is employed as a single source of subcommand buffers. IKJPTGT may obtain its subcommand buffers from two sources:
 1. Subcommand buffers may be supplied from a CLIST.

When CLIST processing is active, the call from BLSUMON to IKJPTGT causes CLIST processing to resume. Any number of command procedure

statements may be processed within a single call to IKJPTGT. Only when a buffer is encountered which does not contain a command procedure statement is that buffer returned to BLSUMON for processing as an IPCS subcommand.

When a CLIST is active, any mode message supplied by BLSUMON is ignored.

2. Subcommand buffers may be supplied from the terminal.

If the subcommand is obtained from the terminal, and BLSUMON has furnished a mode message, the mode message is displayed before soliciting the subcommand. BLSUMON supplies no mode message if the NOMODE option of the TSO PROFILE command has been selected and no special circumstance warrants the display of a mode message.

BLSUMON intercepts those subcommand buffers containing no verb, those in which syntax errors are detected by the TSO command scan service routine (IKJSCAN), and those which contain the END verb. Subcommand buffers containing no verb are treated differently, depending on the context in which they are encountered:

- If the buffer was supplied from the terminal and the DSPL3270 subcommand has suspended operation, the buffer containing no verb is discarded and one containing "DSPL3270" is substituted. (A special mode message is employed when DSPL3270 has suspended operation. In this situation the NOMODE option of the TSO PROFILE command is ignored and a mode message is always supplied.)
- In all other circumstances, the buffer is treated as a null subcommand, and another subcommand buffer is obtained.

Any verb which is not intercepted by BLSUMON is passed to BLSUMONL for general subcommand processing.

Terminal Attention Processing

BLSUMONI receives control when the IPCS user generates a terminal attention interruption. If the DSPL3270 subcommand (BLSR3270) is active, the screen is cleared to ensure that the user can see the dialog initiated by the attention interrupt, and the TSO terminal I/O controller is informed that full-screen mode processing should be terminated. (See the DSPL3270 subcommand for a more extensive discussion of full-screen mode processing.) The dialog with BLSUMONI is initiated by displaying the

IPCS*

mode message. A response is then requested from the IPCS user. The response may elicit five distinct actions from BLSUMONI:

1. If a command containing no verb is entered, BLSUMONI terminates the attention dialog and permits IPCS processing to resume from the point of suspension. (The TSO attention processing mechanism does not terminate active processing when a terminal attention is generated. It suspends active processing.) If the DSPL3270 subcommand is active, the TSO terminal I/O controller is informed that full-screen mode processing should be resumed.

2. If the TSO command scan service routine detects a syntax error, the error is diagnosed, interrupted processing remains suspended, the "IPCS" mode message is displayed again, and another response is requested.
3. If the TIME verb is entered, a time stamp is displayed (IKJEFT25), interrupted processing remains suspended, the
 IPCS*
 mode message is displayed again, and another response is requested.
4. If the ABEND verb is entered, an immediate request for abnormal termination is made. Generation of a dump is requested.
5. If any other verb is entered, the subcommand buffer is queued for processing by BLSUMON and a flag (ZZ2EVEP) is set to request rapid termination of active IPCS processing. The dialog is ended and suspended processing is resumed. The processing is permitted to continue until a reasonable point of termination is reached.

General Subcommand Processing

BLSUMONL employs the description of the subcommand furnished by the TSO command scan service routine (IKJSCAN) and the IPCS subcommand table (as shown in Figure 6) to categorize the subcommand, determining which monitor routine is to provide further processing.

Command	Module	Properties
ADDPROB	BLSEADPR	IPCS, PDR, DSD
COMPARE	BLSRCOMP	IPCS, DUMP
etc		.
.		.
.		.

Properties

- Command Type (IPCS, TSO, etc.).
- Problem Directory required for command usage (PDR).
- Data set directory required for command usage.
- Dump required for command usage.
- Flag indicating command as being not supported under IPCS.

Figure 6. IPCS Subcommand Table

- If extended implicit notation (%verb) has been used to request initiation of a CLIST, BLSUMONC is selected to continue subcommand processing.
- If the verb entered does not match any listed in the IPCS subcommand table, BLSUMONX is selected to continue subcommand processing.
- If the verb is listed as one not supported by IPCS, processing of the verb is terminated, and a user ABEND 16 is simulated.

- If the verb is listed as an IPCS subcommand, BLSUMONL ensures that the subcommand receives control in a suitable environment:
 - The active task variable is refreshed so that it reflects the current SETDEF defaults (BLSUZZ2R).
 - Access to central data base and message queuing services is provided by loading entry point BLSDVT, an alias of load module BLSDVECT associated with data area DVT, and storing its address in ZZ2DVTP. This is done both for subcommands which require access to the problem directory and those which require access to the data set directory.
 - Current allocation of the problem directory is ensured (BLSDAPDR) for those subcommands which require it as indicated by the subcommand table.
 - Current allocation of the data set directory is ensured (BLSDADSD) for those subcommands which require it as indicated by the subcommand table.
 - Access to problem analysis services is provided by loading entry point BLSRVT, an alias of load module BLSRVECT associated with data area RVT, and storing its address in ZZ2RVTP. This is done both for subcommands which require access to dump data.
 - Access to general-purpose dump formatting services is provided by loading entry point BLSTVT, an alias of load module BLSTVECT associated with data area TVT, and storing its address in ZZ2TVTP. This is done both for subcommands which require access to dump data.

If environmental requirements can be satisfied, control is passed to the module which processes the subcommand via program management module BLSUPGMS.

- If the verb is listed as a TSO command, BLSUMONT is selected to continue subcommand processing.

Four monitor routines continue subcommand processing and ultimately transfer control to subcommand processing entry points. These four routines are described below:

BLSUMONT	identifies the entry point concerned with a TSO command verb as the program whose name matches the verb. BLSUMONA is used to ATTACH the command processor.
BLSUMONA	attaches a TSO command processor and supervises the command processor until completion. Standard TSO conventions are employed with regard to subpools shared (78), notification that early termination has been requested (DETACH with the STAE option), and marking data sets no longer in use.
BLSUMONX	uses BLSUBLDL to locate an entry point having the same name as the verb in the subcommand buffer. If such an entry point is found, BLSUMONT is used to pass control to it as a TSO command processor. If no entry point is found, BLSUMONC is used to pass control to the TSO EXEC command processor for "Implicit EXEC" processing.
BLSUMONC	identifies the entry point concerned with CLIST initiation as the TSO EXEC command processor and manipulates the subcommand buffer to indicate that "Implicit EXEC" processing is required. BLSUMONA is used to ATTACH the EXEC command processor.

Related Subcommands

Several subcommands are closely related to IPCS monitor subcommand processing. These subcommands are END, HELP, and TSO.

The END subcommand is intercepted in BLSUMON. BLSUMON initiates IPCS session termination when the END subcommand is encountered.

The HELP (BLSUHELP) and TSO (BLSUTSO) subcommands share common resources with the monitor; each accesses the subcommand table, and each passes control to a TSO command processor via BLSUMONA to provide the service requested by the subcommand.

Storage Management

As shown in Figure 7, automatic storage for most IPCS modules is managed by a single module, BLSUALLO.

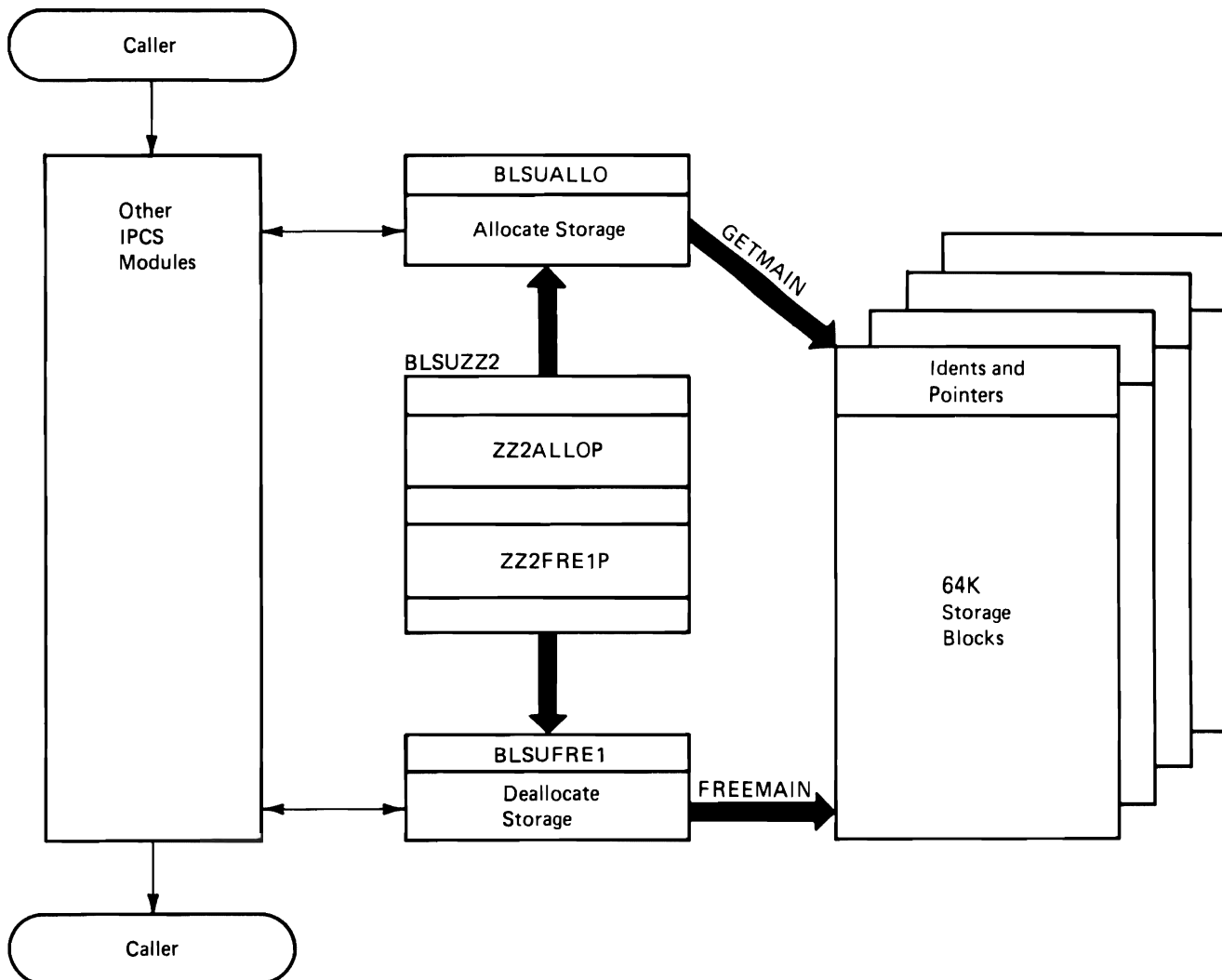


Figure 7. IPCS Storage Management

This module has two entry points:

BLSUALLO Allocate automatic storage
BLSUFRE1 Free automatic storage

This routine obtains virtual storage in 64K byte blocks from subpool 1. Each new request is allocated from this storage pool and a GETMAIN is performed only if sufficient storage is not available. Each IPCS module is allocated automatic storage from this pool during the module entry processing, and the storage is returned to the pool during the module exit processing. All access to the storage pool is via the active task variable (see “BLSUZZ2” on page 406). Subpool usage is summarized below:

SP Use of the Subpool

- 0 Used by cleanup routines which may be entered during ABEND processing for the automatic storage they require.
- 1 Standard TSO usage. This includes most automatic storage, unique buffers obtained (via GETMAIN) by IPCS routines, and buffers returned by TSO service routines.
- 78 Storage shared between the IPCS tasks, including the following:
 - BLSUZZ2 control blocks
 - BLSUVSFB control block
 - BLSRZZ6 control block and I/O buffers for dumps
 - BLDL table
 - DMCBs and related data access service areas
 - Standard TSO shared data

Program Management

The program management routines handle:

- The transfer of control to nonresident routines (BLSUPGMC, BLSUPGME, BLSUPGMR, and BLSUPGMS).
- The loading of nonresident routines into storage (modules BLSUPGML and BLSUPGMU).
- The creation, maintenance, and deletion of a table that lists all the routines that are resident (modules BLSUPGML and BLSUPGMD).
- The retrieval, maintenance, and deletion of a BLDL table for program entry points (modules BLSUBLDL and BLSUBLDD).

These routines bring into storage nonresident programs required for IPCS processing. This includes the loading of IPCS subcommand processors (but is not limited to those routines).

The IPCS task load table, which is an entry point table for nonresident programs, records which programs have been loaded into storage. Since nonresident programs are not loaded until they are needed, the table provides a means of knowing whether or not it is necessary to call the program that will get the program loaded, or simply provide the address of the already loaded routine.

A second table used in the program management process is the BLDL table (BLSUBLDT). It retains the results of BLDL operations for entry points during

the IPCS session, permitting later BLDL requests for the same entry points to be satisfied from virtual storage. It can be expanded as new entries are added to it. Routines BLSUBLDL and BLSUBLDD are used to maintain and delete this table.

When an IPCS session is terminated, the two tables mentioned above are deleted and the storage is returned to the system.

IPCS Subcommand Processing

The modules that process IPCS subcommands share some common attributes.

1. *Initialization*

The monitor (modules BLSUMONx) uses the BLSUPGMR service routine to load and call all the modules, passing the address of the active task variable (see “BLSUZZ2” on page 406) in register 1. This active task variable contains the address of the command line buffer. BLSUZZ2 also establishes addressability to all IPCS common service routines and data areas.

The active task variable contains a field, ZZ2A, that is reinitialized by the monitor prior to the execution of every IPCS subcommand. This area contains, among other things, the SETDEF options in effect during the execution of the subcommand. The monitor reinitializes ZZ2A with information it obtains from the corresponding area of the master task variable, which contains the default options as set by SETDEF. The ZZ2A field might also be refreshed by subcommands that accept the DEFAULT keyword. In this way, the options set in this subcommand do not affect the execution of other subcommands.

2. *Module Format*

Subcommand modules generally consist of at least two CSECTs. The first contains the mainline processing of the subcommand. The second contains the parse control list (PCL) that describes the syntax of the subcommand in the form needed by IKJPARS. Parse validity-check routines are among the remaining CSECTs.

3. *Parsing*

Subcommands parse the command line (BLSUPARI is the IPCS interface to IKJPARS). The command line is compared with the descriptions of the required and optional operands. If invalid operands are discovered or if required operands are missing, diagnostics are issued and prompting might occur. IKJPARS calls the IPCS validity check routines to provide additional validity checking, which might include some semantic checking.

The result of the parse is a parameter descriptor list (PDL). It consists of one or more chained blocks of storage from subpool 1. Each block might contain one or more parameter descriptor entries (PDEs). There is one PDE in the PDL for each possible operand. The PDE depends on the type of operand, but might indicate such things as whether the operand is present, what keyword of a set was specified, etc. (See *TSO Guide to Writing a TMP or a Command Processor* for a description of IKJPARS, PCLs, and PDLs.)

The PDL storage becomes the property of the subcommand. The subcommand releases the PDL storage prior to exiting. A register is used within the subcommand to establish addressability to the PDL.

After the subcommand is parsed, the PDE for the TEST and NOTEST keywords is examined. If either of the keywords is present, the corresponding flag in the IPCS active task variable (see "BLSUZZ2" on page 406) is overridden.

4. Common Subroutines

IPCS processing uses common subroutines to process PDEs representing groups of operands common to multiple subcommands:

Routine	Service Provided
BLSRADDR	Accepts the PDEs for address processing operands. Updates the current task variable (see "BLSUZZ2" on page 406) to indicate the source of data for this subcommand, performs dump initialization to prepare the dump source for use, and returns an equate symbol record (see "BLSRESSY" on page 373) describing the storage within that dump source.
BLSRADD5	Performs the same services as BLSRADDR. Also updates symbol X, which represents the current address space, to describe the storage.
BLSRDUSE	Accepts the PDEs for a data source. Updates the current task variable (see "BLSUZZ2" on page 406) to indicate the source of data for this subcommand.
BLSRDUSO	Performs the same services as BLSRDUSE. Also performs dump initialization to prepare the dump source for use.
BLSRPADS	Accepts the PDEs for the DISPLAY keyword and options. Updates the current task variable (see "BLSUZZ2" on page 406) to indicate the options in effect for this subcommand.
BLSRVAL	Accepts a PDE for a general value. Returns a value resolution buffer (see "BLSRVALY" on page 396) describing the value.
BLSUPARU	Accepts the PDEs for message routing and the FLAG keyword with options. Updates the current task variable (see "BLSUZZ2" on page 406) to indicate the options in effect for this subcommand.
BLSUSIGR	Accepts the PDEs for a range of signed integers. Returns two fullword signed integers.

5. Mainline and Clean-Up Processing

During mainline processing each subcommand references entries in the PDL and BLSUZZ2 as necessary.

When subcommand processing is complete, the PDL storage is released (FREEMAIN) and control is returned to the monitor.

IPCS Session Control Commands and Subcommands

The IPCS session control subcommands terminate the IPCS session (END), control default values during the session (SETDEF), make reference information available regarding subcommands (HELP), and provide access to TSO commands whose names duplicate the names of IPCS subcommands (TSO). The COPYDUMP subcommand is used to selectively transcribe dump data sets from one storage device to another.

SETDEF Subcommand

The SETDEF subcommand is entered at entry point BLSRSETD in load module BLSRSETD.

BLSUPARI is called to parse the operands of the SETDEF subcommand and return a tabular description of the operands if parsing is successful. If an error is detected and cannot be resolved during parse processing, SETDEF immediately terminates.

Field ZZ2A in the active task variable (see “BLSUZZ2” on page 406) is modified to reflect any changes requested. As the individual changes are processed, additional checking is performed:

- The PRINT option is rejected if FILE(IPCSPRNT) is not open.
- The default CPU address is rejected unless it makes sense for the default data set.
- The default ASID is rejected unless it makes sense for the default data set.

If all requested updates are acceptable, the field ZZ2A in the master task variable is updated.

If the LIST option was requested or implied, the current defaults are displayed.

HELP Subcommand

BLSUHELP invokes the TSO HELP command processor to display information about the command entered. BLSUHELP checks the IPCS subcommand table to see if the command about which information is requested is an IPCS subcommand. If it is an IPCS subcommand, the help text is retrieved from the IPCS member of the SYS1.HELP data set. Otherwise, the HELP text is retrieved from the specified member of the standard TSO HELP data set.

TSO Subcommand

BLSUTSO processing of the TSO subcommand depends on one of the following situations:

- If a TSO command is not specified, BLSUTSO calls BLSUMOAT to issue an ATTACH macro for module BLS9INI2. BLS9INI2 establishes a subcommand table for TSO subcommand processing (BLS9STBL), establishes a terminal attention exit to receive control during TSO subcommand processing (BLS9MONI), and calls BLSUMON to solicit TSO commands and CLISTs until an END subcommand is entered.
- If a TSO command is specified and IPCS processing is being performed interactively (not a background job), BLSUTSO calls BLSUMONT to issue an ATTACH macro for module BLS9CMD1 (via BLSUMONA). BLS9CMD1 establishes a subcommand table for TSO subcommand processing (BLS9STBL), establishes a terminal attention exit to receive control during TSO subcommand processing (BLS9MONI), and calls BLSUMONA to solicit TSO commands and CLISTs until an END subcommand is entered.
- If a TSO command is specified and IPCS processing is being performed in a background job, BLSUTSO calls BLSUMONT to issue an ATTACH macro for the module of the designated TSO command (via BLSUMONA). BLSUMONA does not accept any CLIST invocations.

END Subcommand

There is no IPCS code that executes the END subcommand. Instead, the END subcommand is intercepted by the monitor (BLSUMON), and termination processing is performed.

COPYDUMP Subcommand

BLSRCOPY is the module that processes the COPYDUMP subcommand, as shown in Figure 8 on page 31.

BLSRCOPY copies records from an input data set and writes them to a specified target (or output) data set. Both the input and target data sets may be allocated via the COPYDUMP subcommand. If the user chooses to allocate either or both data sets prior to entering the COPYDUMP subcommand, BLSRCOPY performs the open only of the specified data sets.

BLSRCOPY first parses, via IKJPARS, the keywords and sets up all flags used during execution. Next, a check must be made to see if the input data set has already been allocated by COPYDUMP during a previous execution. If so, and the DMCB indicates that the dump is still allocated and open, then data set allocation for the input data set is bypassed. If no DMCB exists for the specified data set, BLSRCOIN is called to allocate and open the input data set. If SKIP(EOF) is specified, no target data set allocation takes place.

On each entry to BLSRCOPY it must be determined whether the dump copying process is to resume at the point in the dump where it left off during the last execution, or whether processing should start from the beginning of the input data set. To do this BLSRCOPY obtains storage and places the CCB (a work data area containing flags, pointers, dsname, ddname, etc.) in it. The CCB is freed and the input data set closed and deallocated when any one of the following conditions occurs:

- EOF is reached
- The CLOSE keyword is specified
- The IPCS session is terminated
- The subtask ABENDs

After the input data set has been allocated and opened, BLSRCOPY calls BLSRCOTA to allocate the target data set. A check is also made to see if there are any records in the IPCS dump directory that are associated with the target data set and, if so, they are deleted from the directory.

If both the input and target data sets are successfully allocated and opened, BLSRCOPY enters a loop, reading each successive input data set dump record. BLSRCOHD is called to process any dump header records that are encountered. The loop is exited when the action requested on the command line is completed. That action may be the transcription of a dump, the skipping over of n dumps and then the copying of the next dump, or the skipping over of all the dumps in the data set while displaying their titles, as in the case of SKIP(EOF).

When all processing has been completed, the input data set is closed and deallocated (if allocation was done by BLSRCOPY) unless the user specifies that it is to be left open. The target data set is always closed by BLSRCOPY (and deallocated if allocated by BLSRCOPY). If the user specifies the DEFAULT option, then the target data set is made the IPCS current dump data set (unless it was specified by ddname).

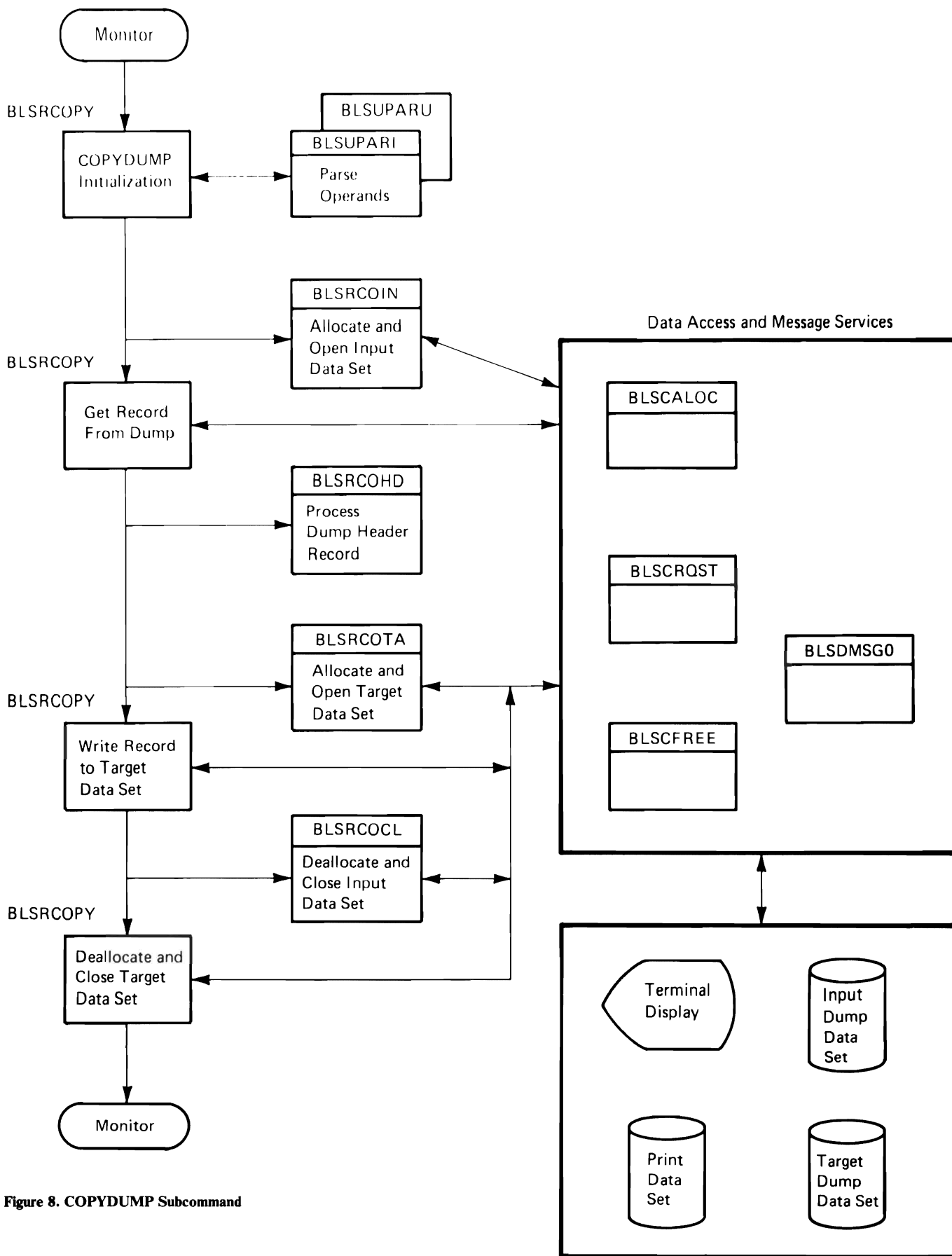


Figure 8. COPYDUMP Subcommand

Problem and Data Management Subcommands

The problem and data management subcommands enable the user to keep track of problem information, particularly the data sets associated with problems. These subcommands are used to add, modify, and delete problems and data sets, as well as provide lists of this information.

ADDPROB Subcommand

The entry point for processing the ADDPROB subcommand is BLSEADPR. As shown in Figure 9 on page 33, BLSEADPR calls BLSUPARI to parse the subcommand parameters, and calls BLSDDTIME to get the current date and time. It opens the PDR for output (update) and verifies the problem number to be added. The problem number is determined in one of two ways, depending on the presence of the PROBLEM keyword:

1. If the PROBLEM keyword is present, the value supplied with it is used for the problem number. An attempt is made to point to the status record of the specified problem. A successful point means that the problem already exists in the PDR; an error message is issued, and the subcommand is terminated with a return code of 12. An unsuccessful point indicates that the problem does not exist, and processing continues. The PDR seed value is not updated when the PROBLEM keyword is present.
2. If the PROBLEM keyword is not present, the seed record is read from the PDR. The problem-id value from the seed record is incremented until an unused problem number is detected; once found, the updated value is written back to the PDR. The new seed record value is the number of the problem about to be created; thus, the seed value record contains the last problem number assigned by IPCS.

BLSEADPR builds a problem status record for the new problem using the parameters supplied on the ADDPROB subcommand, and assigns default values for each field that was unspecified. If the problem description is in a data set (identified by the use of the DSDESCRIPTION keyword), BLSEADPR calls BLSEADP1 to copy the problem description into the PDR. If the problem description is supplied with the subcommand as a text string (identified by the use of the DESCRIPTION keyword), the description records are generated directly by BLSEADPR.

The new problem status record is written to the PDR and the PDR is closed. If the procedure was completed successfully, the user is informed of the problem number assigned via message BLS05100I. If the DEFAULT keyword was present in the subcommand, the new problem number is set into the default problem number field of the IPCS master task variable.

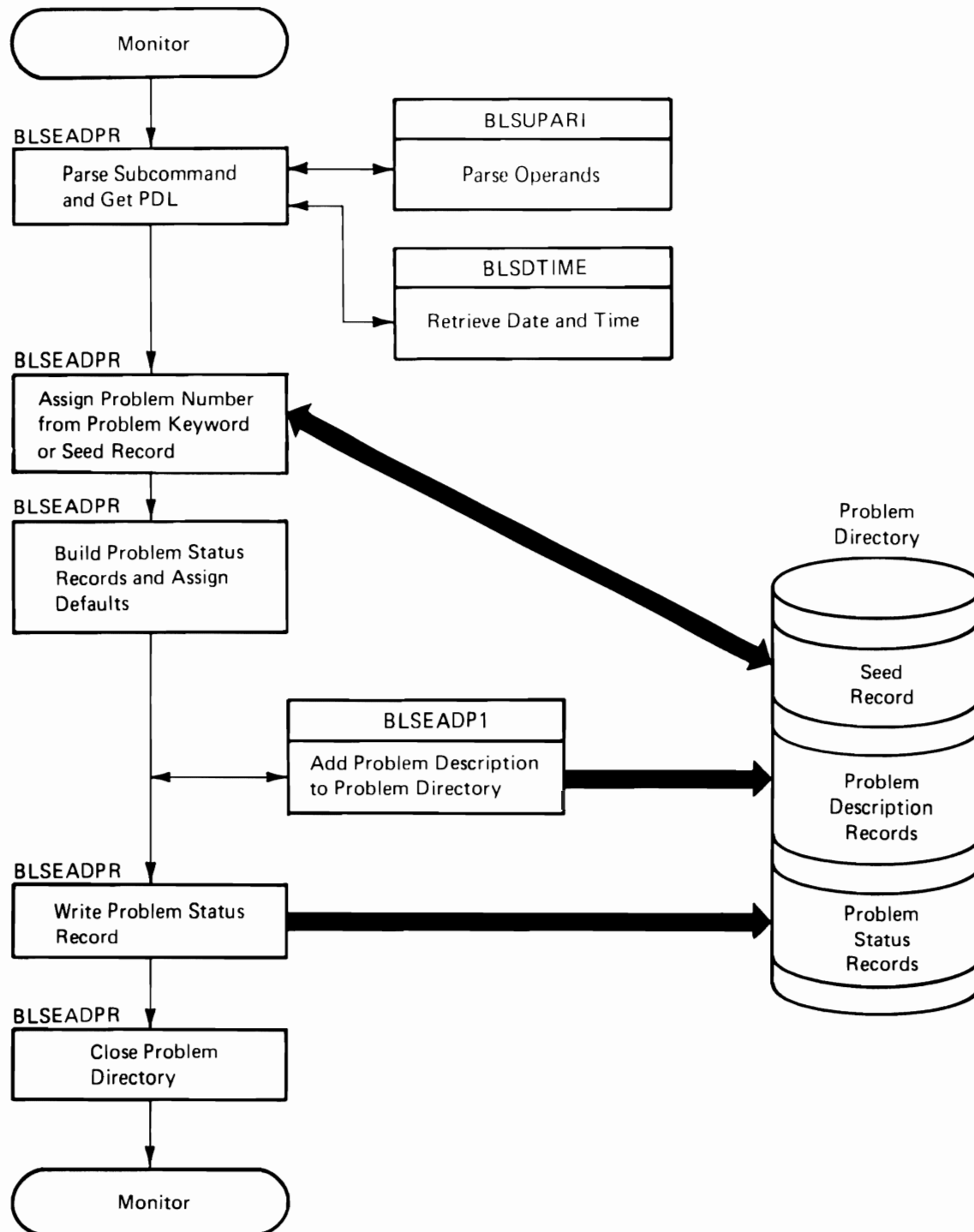


Figure 9. ADDPROB Subcommand

DELPROB Subcommand

The entry point for processing the DELPROB subcommand is BLSEDEL0. As shown in Figure 10 on page 35, BLSEDEL0 calls BLSUPARI to parse the subcommand parameters. BLSEDEL0 constructs the key for the status record of the problem to be deleted. BLSDENQP is called to obtain exclusive use of that problem resource, in order to prevent concurrent access to the problem's data. The PDR is opened, and the problem status record is read from the PDR. If the record is not found, deletion proceeds without confirmation or authorization checking. This continuation cleans up any records pertaining to the problem which may exist in either the PDR or DSD due to a failure during a previous IPCS subcommand.

If the status record is found, BLSEAUTH is called to determine whether the user is authorized to delete the problem. BLSEAUTH does this by comparing the userid field in the protected step control block (PSCB) with the fields in the problem status record and in the IPCS facility parameters control block (FPBLOK).

If the user's authorization is not verified, message BLS05400I is issued and BLSEDEL0 enters its termination processing. If the user's authorization is verified, BLSEDEL1 is called.

If the CONFIRM option is in effect, BLSEDEL1 calls BLSEDELC. BLSEDELC provides confirmation processing for the DELPROB subcommand by calling BLSELPCL, which constructs a data output queue containing information about the problem and its associated data sets. After the data output queue is displayed and return is made, BLSEDELC calls BLSFCN00 to prompt the user for confirmation via message BLS05402D.

If the user confirms the request by replying Y, BLSEDEL1 calls BLSFDDDP to actually delete the records from the PDR and DSD. (This operation, "DELPROB Connection For DELDSN," is described in the DELDSN subcommand.) If the user does not confirm the request, BLSEDEL1 issues the message BLS05401I and returns to BLSEDEL0.

BLSEDEL0, in order to terminate its processing, calls BLSDDDEQP to release the problem resource it had earlier obtained. BLSEDEL0 then returns control to the IPCS monitor.

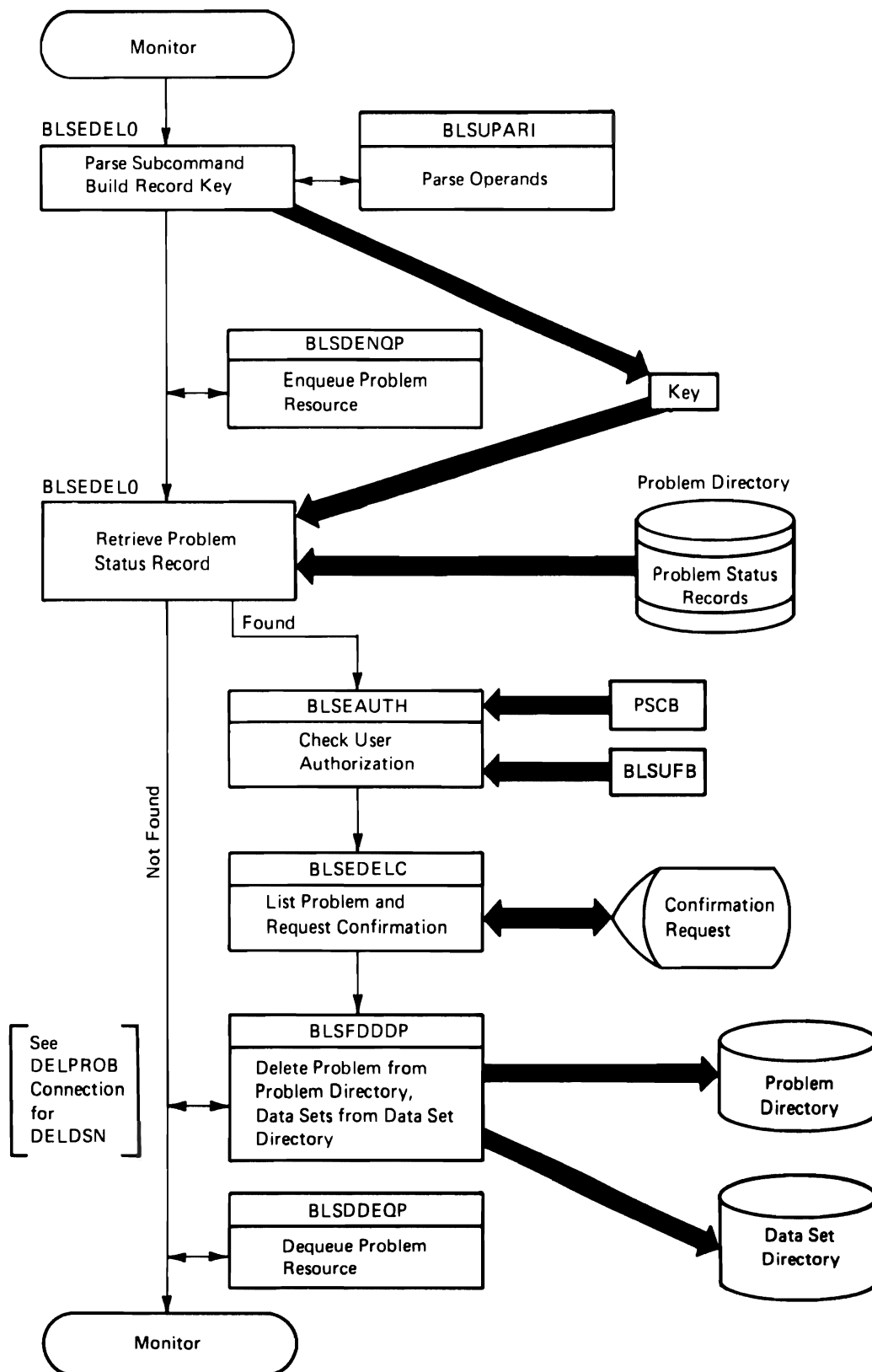


Figure 10. DELPROB Subcommand

LISTPROB Subcommand

As shown in Figure 11 on page 38, BLSELP00 calls BLSUPARI to parse the subcommand parameters. BLSELP00 determines the size of the LISTPROB workarea by scanning the PDEs returned from IKJPARS. The following internal procedures are used for the scan:

CNTLIST	Determines the space needed to represent a list type selection criterion (such as GROUP, SYSTEM, USER, and COMPID).
COUNTKEY	Determines the space needed to represent a keyword type selection criterion (such as PSTATUS, IBMSTATUS, and FIXSTATUS).
CNTOWNR	Determines the space needed to represent the owner selection criterion.
CNTSEV	Determines the space needed to represent the severity selection criterion.

The LISTPROB work area is built by BLSELP00 using the following internal procedures to build selection criterion blocks (SCBs) for the specified operands:

BLDLIST	GROUP, SYSTEM, USER, and COMPID
BLDOWNR	OWNER
BLDSEV	SEVERITY
BLDKEY	PSTATUS, IBMSTATUS, PTFSTATUS, and FIXSTATUS

These SCBs are chained, each SCB containing the sorted list of values for that field, and the offset and length of that field's location within the problem status record of the PDR. The LISTPROB iteration driver (BLSELPDR) is called. BLSELPDR opens the PDR, and obtains the current time of day so that the elapsed time the PDR is open can be monitored. The following conditions determine the processing which BLSELPDR performs:

1. If the user requested all problems through the ALL keyword, or if he specified selection keywords but not the PROBLEMS keyword, the following are performed for each problem status record in the PDR:
 - a. Fetch a problem status record, which contains all the problem data except the description and the data set associations.
 - b. Call BLSELPCL.
2. If the user did not specify either the PROBLEM keyword or any selection keywords on the subcommand:
 - a. Fetch the problem status record for the default problem.
 - b. Call BLSELPCL.
3. If the user specified a list of problems and/or ranges:
 - a. Fetch the problem status record that represents the next problem in the list or in the range.
 - b. Call BLSELPCL.

BLSELPCL calls BLSELPCC, which scans the SCB chain. For each element in the SCB chain, the value in the problem status record is used as the key for a binary search of that SCB element's values. If that value is found, the next element in the SCB chain is located, and the process is repeated. If that value is not found, BLSELPCC indicates via its return code that the problem is not to be included in the output report. In this way, BLSELPCC determines whether the current problem meets the criteria specified by the user through his selection keywords. The problem meets the selection criteria of a keyword if the problem's value for the field named by the keyword is equal to any of the values specified in the keyword on the subcommand. Control is returned to BLSELPCL.

If the problem is to be included in the output report, BLSELPCL calls BLSELPPOP. BLSELPPOP performs the following steps:

1. BLSDMSG0 is called to build a data output queue element(s) according to the report format specified by the user through the output keywords.
2. If the DESCRIPTION option was specified by the user, the description records for the current problem are retrieved from the PDR and put onto the data output queue.
3. If the DSNAMES option was specified by the user, the data set association records for the current problem are fetched from the PDR, and BLSFLP00 is called to obtain the data set information from the DSD and to put it onto the data output queue.

Control is returned to BLSELPCL. BLSELPCL calls BLSELPCT, which determines whether the PDR resource has been in use for an excessive length of time. The following checks are performed:

1. Determine whether the PDR has been open more than the allowable amount of time by obtaining the current time and comparing it against the time at which the PDR was last opened.
2. Determine whether a maximum number of lines are in the data output queue.
3. Determine whether the problem being processed is the default problem.

If any of these three checks are true, the PDR is closed, thereby releasing the PDR resource. While the PDR is closed, the contents of the data output queue are written by calling BLSDMSG0. Upon return, the PDR is again opened, and the current time of opening is saved. Control is returned to BLSELPCL.

If none of these three checks is true, control is returned to BLSELPCL.

BLSELPCL returns control to BLSELPDR. BLSELPDR determines, according to the conditions listed above, whether another problem status record is to be retrieved from the PDR.

If no more problem status records are to be retrieved, BLSELPDR closes the PDR, and writes the contents of the data output queue by calling BLSDMSG0. BLSELPDR then writes a message indicating the number of problems that were listed. BLSELPDR returns control to BLSELP00, which frees the LISTPROB work area (containing the SCBs) and returns control to the IPCS monitor.

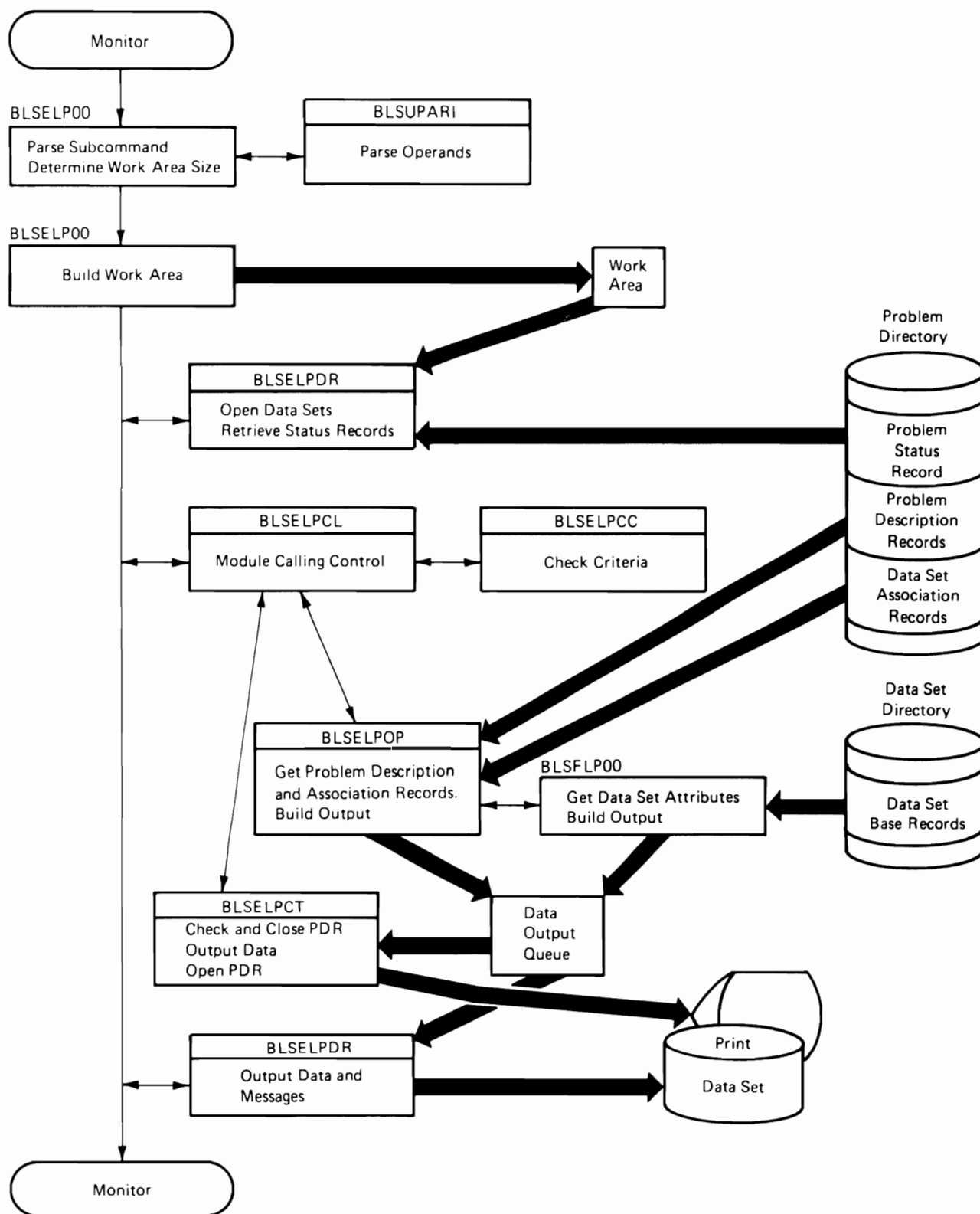


Figure 11. LISTPROB Subcommand

MODPROB Subcommand

The entry point for processing the MODPROB subcommand is BLSEMDPR. As shown in Figure 12 on page 40, BLSEMDPR calls BLSUPARI to parse the subcommand parameters, and calls BLSDTIME to get the current date and time. BLSEMDPR obtains the problem number that was specified on the subcommand or, if it wasn't specified, it obtains the default from the active task variable (ATV). If neither has been specified, BLSEMDPR issues message BLS04003I and terminates processing by returning to the IPCS monitor with a severe error return code.

Using the problem number, BLSEMDPR opens the PDR, reads the problem status record for the problem, and closes the PDR. If the status record for the problem is not found, message BLS04000I is issued and processing is terminated. If the status record is read successfully, BLSEMDPR calls BLSDENQP to exclusively enqueue on the problem resource. It calls BLSEAUTH to test the user's authority to modify the problem. If the user is not authorized, and modification of the problem status record is attempted, BLSEMDPR issues message BLS04004I and inhibits modification of the problem status record.

If the user is authorized to modify the problem, the values specified on the subcommand are placed into their respective fields of the problem status record. If the DESCRIPTION keyword was specified, the text is appended to the current problem description; this step is performed independent of the user's authority. The last problem description record, if it is not full, is read, and text is transferred to it. When it becomes full, or if it was already full, one or more new problem description records are generated until the amount of text specified is exhausted. As the problem description records become full, they are written to the PDR.

If the DSDESCRIPTION keyword was specified and the user is authorized, BLSEMDPR calls BLSEMDP1. BLSEMDP1 first erases all the problem description records for the problem. It then calls BLSEADP1, which reads the new description from the specified data set, transfers the text into a problem description record and, when full, writes the problem description record(s) into the PDR. Return is made to BLSEMDP1, which returns to BLSEMDPR.

BLSEMDPR writes the updated problem status record into the PDR and closes it. BLSEMDPR calls BLSDDEQP to dequeue the problem resource. If the DEFAULT keyword was present in the subcommand, and no errors occurred, the problem number is set into the default problem number field in the IPCS master task variable. BLSEMDPR then returns control to the IPCS monitor.



ADDDSN Subcommand

The entry point for processing the ADDDSN subcommand is BLSFAD00. BLSFAD00 calls BLSUPARI to parse the subcommand parameters, and BLSFAV00 to verify these parameters. Figure 13 on page 43 shows the operation of the ADDDSN subcommand.

BLSFAV00 calls BLSFPS00 to build the access key for the problem status record, BLSDENQP to exclusively enqueue on the problem resource, and BLSFGP00 to retrieve the problem status record from the PDR, using the key just generated. BLSFDS00 is called to construct the access key for the data set base record (if any should already exist) for the data set being added. BLSFAV00 determines whether the data set being added is a member of a partitioned data set and, if it is and if the MANAGED attribute is specified, forces that attribute to UNMANAGED.

BLSFAD00 calls BLSFGD00 to retrieve the data set base record from the DSD, using the key generated above. If the data set base record is retrieved, the data set is already known to IPCS. BLSFFP00 is called to determine whether the data set is already associated with the problem. BLSFFP00 calls BLSFGD00 to retrieve the problem association record from the DSD, and BLSFGP00 to retrieve the data set association record from the PDR. If both records are successfully retrieved, BLSFAD00 issues message BLS04083I and enters its termination processing. If neither record is retrieved, the association between the problem and the data set is performed as described below. If only the problem association record is retrieved, it indicates that an incomplete association exists due to an interruption to a previous ADDDSN or DELDSN subcommand. In this case, only the data set association record is built and added to the PDR in the process described below. The case of the data set association record existing without the problem association record existing cannot occur, because of the sequence in which the records are added to (ADDDSN) or erased from (DELDN) the DSD and PDR data sets.

BLSFAD00 now performs the association function. BLSFND00 is called to build a new data set base record from the keywords (or their defaults) specified in the subcommand. BLSFOD00 is called to compare the new base record with the old base record, if any. If an old base record exists, and if an attribute in the new base record (which was specified on the subcommand, not a default value) conflicts with that attribute in the old base record, BLSFOD00 calls BLSFDF00 to construct the confirmation messages, and calls BLSFCN00 to request confirmation from the user. The result of the user reply is returned to BLSFAD00. If the user replies N, BLSFAD00 enters its termination processing. If the user replies Y, or if no conflicts exist, the association process continues.

If there was no old data set base record, that is, if the data set name is new to IPCS, BLSFAD00 calls BLSFUD00 to write the new base record into the DSD.

BLSFAD00 calls BLSFBP00 and BLSFND01 to construct the data set association record and the problem association record respectively. The next data set association record sequence number is updated in the problem status record, and BLSFUP00 is called to rewrite the problem status record into the PDR. BLSFUD00 is called to write the generated problem association record into the DSD, and BLSFUP00 is called to write the data set association record into the PDR.

If no errors have been detected and if the DEFAULT keyword was specified on the subcommand, BLSFAD00 calls BLSFSD00 to establish the new default value. BLSFSD00 calls BLSRDUCC to close and free the old data set; it then copies the new data set name into the default data set name field of the IPCS master task variable.

BLSFAD00 termination processing consists of calling BLSDDEQP to release the problem resource, and freeing the parameter descriptor list storage. Control is then returned to the IPCS monitor.

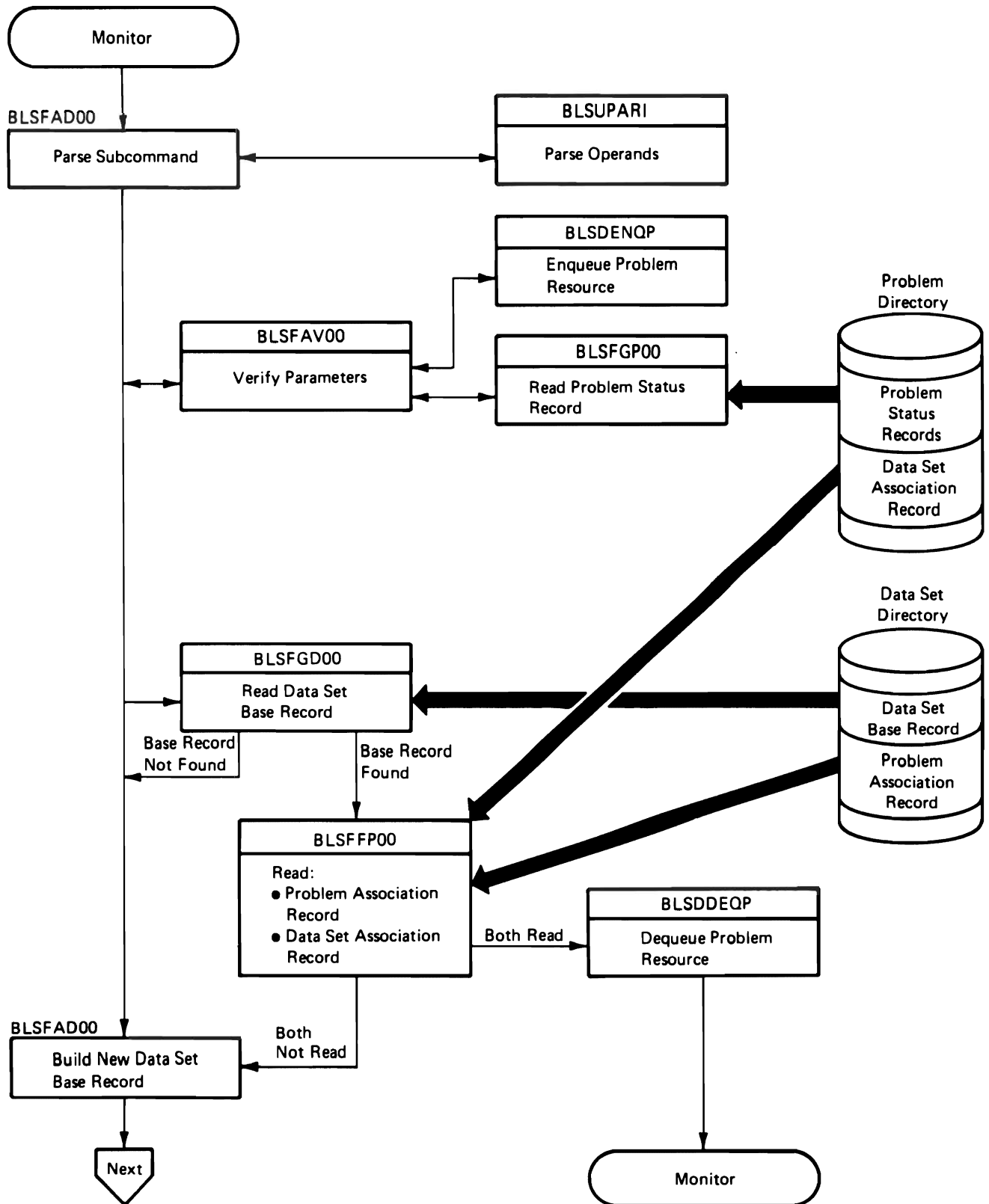


Figure 13 (Part 1 of 2). ADDDSN Subcommand

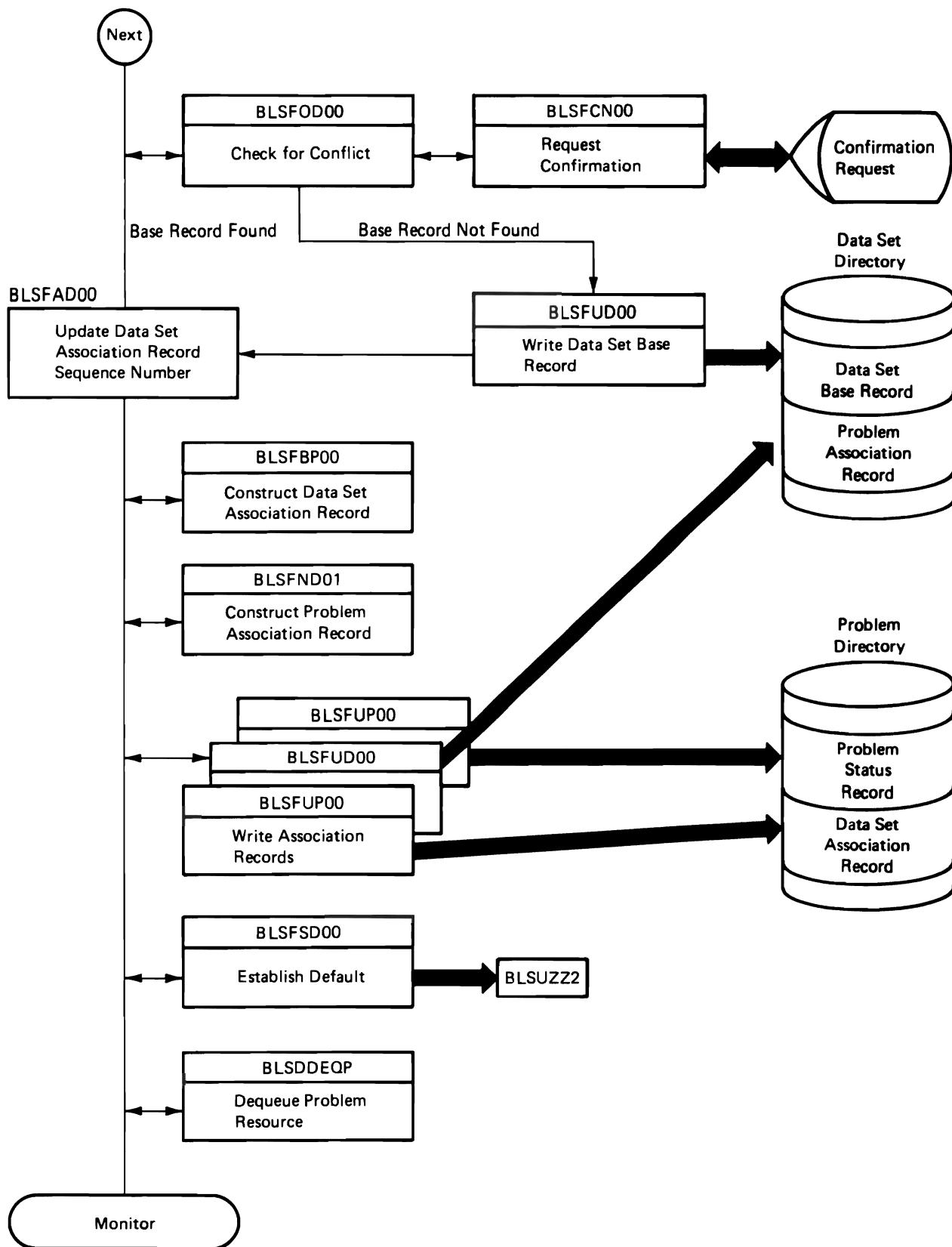


Figure 13 (Part 2 of 2). ADDDSN Subcommand

DELDSN Subcommand

As shown in Figure 14 on page 47, the entry point for processing the DELDSN subcommand is BLSFDD00. BLSFDD00 calls BLSFPARS, which calls BLSUPARI to parse the subcommand parameters. These parameters, or their defaults if unspecified, are placed into variables by BLSFPARS.

BLSFDD00 calls BLSDENQP to exclusively enqueue on the problem resource to guarantee that the problem remains static during the update process.

BLSFDD00 calls BLSFVRFY, which opens the PDR, retrieves the problem status record, closes the PDR, and calls BLSEAUTH to verify that the user has authorization to perform the DELDSN subcommand. If the user does not have authorization, BLSFVRFY issues message BLS04004I and, upon return, BLSFDD00 performs its termination processing.

At this point, an association list is built. The association list is a linked list in which each element contains a problem identifier and a data set name which are associated. Each element is similar to a PDR data set association record, with a link field appended.

If the ALL keyword (instead of the DSNNAME keyword) was specified on the subcommand, BLSFDD00 calls BLSFLALL to build the association list. The association list contains all the data sets associated with the problem. The association list elements are built by BLSFLALL opening the PDR, calling BLSELPFR to read the data set association records connected with the problem, and closing the PDR. The subcommand is terminated if no associations are found.

If the DSNNAME keyword (instead of the ALL keyword) was specified on the subcommand, BLSFDD00 calls BLSFLONE to build an association list of one element. This element is built directly from the data set name and the problem number specified in the command.

If the CONFIRM operand is in effect, BLSFDD00 calls BLSFCONF to provide confirmation processing. BLSFCONF calls BLSELPDS to print the data set names in the association list. BLSELPDS calls BLSFLP00 to print each name. BLSFCONF calls BLSFCN00 to issue the confirmation message BLS05402D and retrieve the reply. An “N” reply terminates the subcommand; a “Y” reply allows processing to continue.

If a positive confirmation is received, BLSFDD00 calls BLSFDALL to delete the associations specified in the association list. BLSFDALL calls BLSFDEL for each element in the list.

BLSFDEL opens the PDR and DSD data sets, and reads the DSD problem association record, thereby obtaining a pointer to the PDR data set association record. If the DSD problem association record was found, then BLSFDEL calls BLSFPRES to erase the data set association record from the PDR, and calls BLSFREAS to erase the problem association record from the DSD. The PDR and DSD data sets are closed. If the data set being dissociated is no longer associated with any problems and is also the default data set, BLSRDUCC is called to close and free it. If the data set is no longer associated with any problems, BLSFDEL calls BLSFDEERS to erase the DSD base record for the data set, and calls BLSFSCRT to scratch or initialize the data set if it had the MANAGED attribute.

After all association list elements have been processed, BLSFDD00 calls BLSELPFM to free the storage occupied by the association list. BLSFDD00 calls BLSDDDEQP to free the problem resource, and returns control to the IPCS monitor.

If the user signals an attention interrupt any time during the processing, the subcommand is terminated. However, if the data base is being updated when the attention is signalled, then termination is deferred until the data base is in a consistent state.

If the command is terminated for any reason, such as an attention interrupt, an 'N' reply to confirmation, no data sets associated with the problem, or the user is not the problem owner, the appropriate cleanup is performed: storage is freed, data sets closed, and the problem is dequeued.

DELPJOB Connection For DELDSN:: The controlling module which supplies the interface from the DELPROB subcommand to the DELDSN function is BLSFDDDP. This is shown in Part 3 of Figure 14 on page 47. It not only dissociates all data sets from the problem, but is also responsible for deleting all records in the PDR representing the problem itself.

BLSFDDDP calls BLSFLALL which, together with BLSELPFR, builds the data set association list. The data set association list is constructed from information contained in the data set association records for the problem in the PDR. If no data set association records are found, the data set association list is empty. BLSFDDDP calls BLSFDALL to delete the associations specified in the association list, if any. The operation of BLSFDALL is described above in the DELDSN subcommand.

After all association list elements have been processed, BLSFDDDP calls BLSELPFM to free the storage occupied by the association list. BLSFDDDP calls BLSFPERA to erase the problem description records and the problem status record from the PDR, thereby deleting the problem. BLSFDDDP then returns control to its caller.



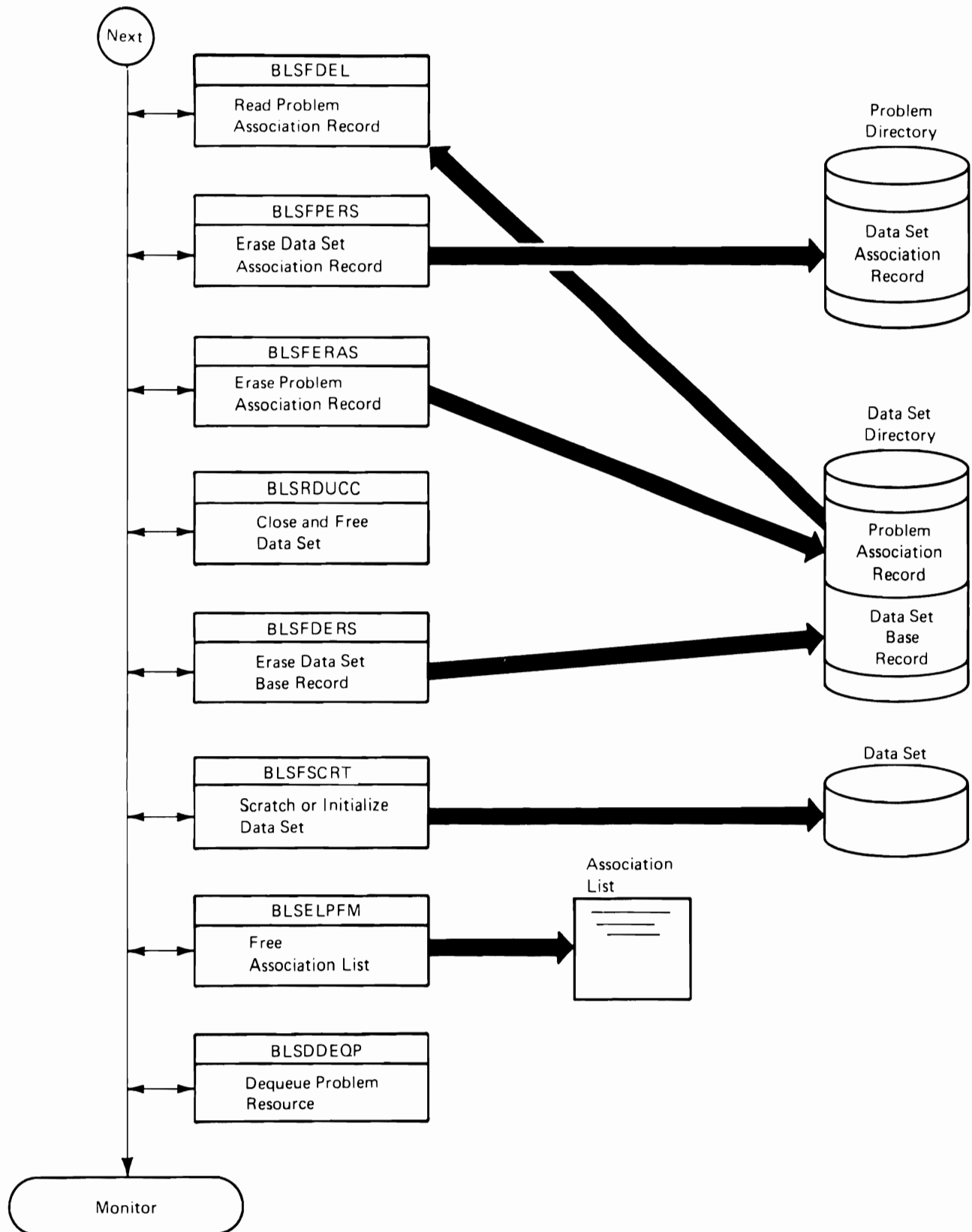


Figure 14 (Part 2 of 3). DELDSN Subcommand

LISTDSN Subcommand

The entry point for processing the LISTDSN subcommand is BLSFLD00. BLSFLD00 calls BLSUPARI to parse the subcommand parameters. Figure 15 on page 51 shows the operation of the LISTDSN subcommand.

If the DSNAME keyword (instead of the ALL keyword) was specified on the subcommand, BLSFLD00 calls BLSFDS00 to construct the key of the data set base record, and calls BLSFGD00 to retrieve the data set base record from the DSD. If the record is successfully read, BLSFLD00 calls BLSFDF00 to format the output report.

BLSFDF00 calls BLSFFL00 to generate the first two lines of the report, consisting of the data set name and the data set's attributes, respectively. It then calls BLSFSL00 to produce the report line which contains the description of the data set. If the PROBLEMS keyword was specified on the subcommand, BLSFDF00 calls BLSFTL00 to produce the report line(s) indicating the problem identifiers with which the data set is associated.

BLSFTL00 generates the key for the next (first) problem association record, calls BLSFGG00 to retrieve it, and places the problem identifier into the output line. BLSFTL00 repeats this process until all problem association records connected with this data set base record have been retrieved.

If the ALL keyword (instead of the DSNAME keyword) was specified on the subcommand, BLSFLD00 instead constructs the key of the first data set base record and calls BLSFGG00 to retrieve it. BLSFLD00 then calls BLSFDF00 to format the output report for that data set base record (described above). Upon return, BLSFLD00 outputs the generated report and generates the key for the next data set base record. The process is repeated until no more data set base records are to be found.

Upon completion, BLSFLD00 returns control to the IPCS monitor.

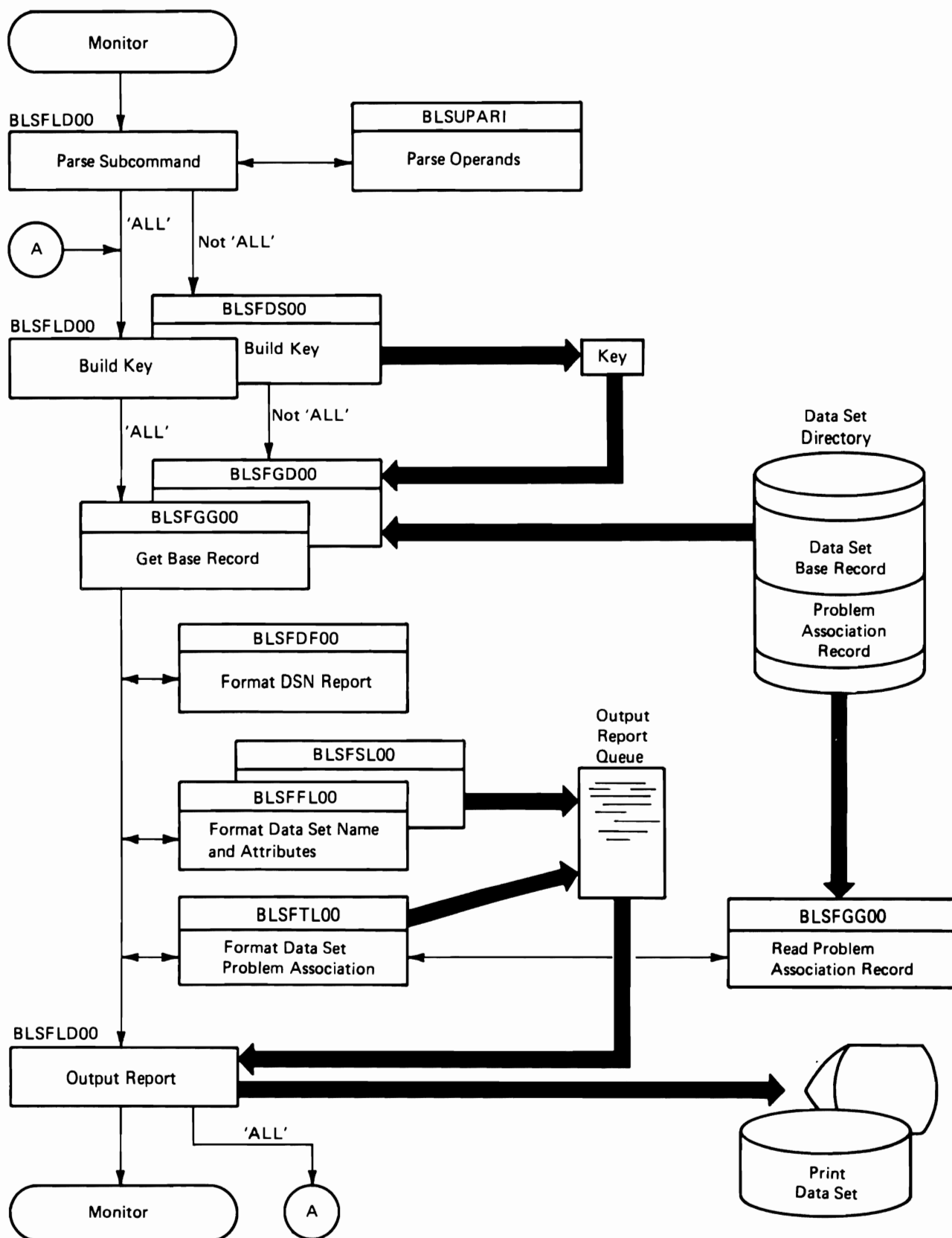


Figure 15. LISTDSN Subcommand

MODDSN Subcommand

The entry point for processing the MODDSN subcommand is BLSFMD00. Figure 16 on page 53 shows the operation of the MODDSN subcommand. BLSFMD00 calls BLSUPARI to parse the subcommand parameters. BLSFDS00 and BLSFPS00 are called to construct the keys of the data set base record and problem status record, respectively. BLSFMD00 checks whether the data set name was specified with a member name and whether the attribute MANAGED was also specified. If so, message BLS04010I is issued and the attribute is forced to UNMANAGED.

BLSFMD00 calls BLSDENQP to exclusively enqueue on the problem resource, to ensure that no other user can obtain it during the update process.

BLSFMD00 calls BLSFGP00 to open the PDR, retrieve the problem status record using the key constructed earlier, and close the PDR. BLSEAUTH is called to ensure that the user has authorization to perform the modification. If he does not have authorization, message BLS04004I is issued and BLSFMD00 performs its termination processing.

If the user does have authorization, BLSFMD00 calls BLSFGD00 to open the DSD, retrieve the data set base record using the key constructed earlier, and close the DSD. BLSFFP00 is called to ensure that the data set name is associated with the problem. BLSFFP00 does this by constructing the key for the problem association record, and calling BLSFGD00 to open the DSD, retrieve the problem association record, and close the DSD. BLSFFP00 constructs the key for the data set association record, and calls BLSFGP00 to open the PDR, retrieve the data set association record, and close the PDR. If both records are successfully read, the problem and data set are associated. If they are not associated, upon return BLSFMD00 issues message BLS04009I and performs termination processing.

BLSFMD00 calls BLSFND00 to construct a new data set base record, based upon the attributes specified on the subcommand. Upon return, BLSFMD00 copies the attributes that weren't specified on the subcommand into the new data set base record from the old data set base record.

BLSFMD00 calls BLSFOD00 to determine whether any data set attribute conflicts exist between the old and the new data set base records. If conflicts exist, and if CONFIRM is in effect, BLSFOD00 calls BLSFDF00 to produce a report (including a list of problems with which the data set is associated), writes this report to the terminal, and calls BLSFCN00 to request confirmation for the update. The result of the user reply is returned to BLSFMD00. If NOCONFIRM is in effect and, if conflicts do exist, message BLS04062I is issued; regardless of whether conflicts exist, BLSFOD00 returns a positive confirmation to BLSFMD00.

If a negative confirmation is received, BLSFMD00 performs termination processing. If a positive confirmation is received, BLSFUD00 is called to open the DSD, replace the old data set base record with the new one, and close the DSD. If no errors have been detected, and if the DEFAULT keyword was specified on the subcommand, BLSFMD00 calls BLSFSD00 to update the default data set name in the master task variable. BLSFSD00 first calls BLSRDUCC to close and free the old default data set, and then performs the update.

BLSFMD00 termination processing calls BLSDDQEP to dequeue the problem resource before returning control to the IPCS monitor.

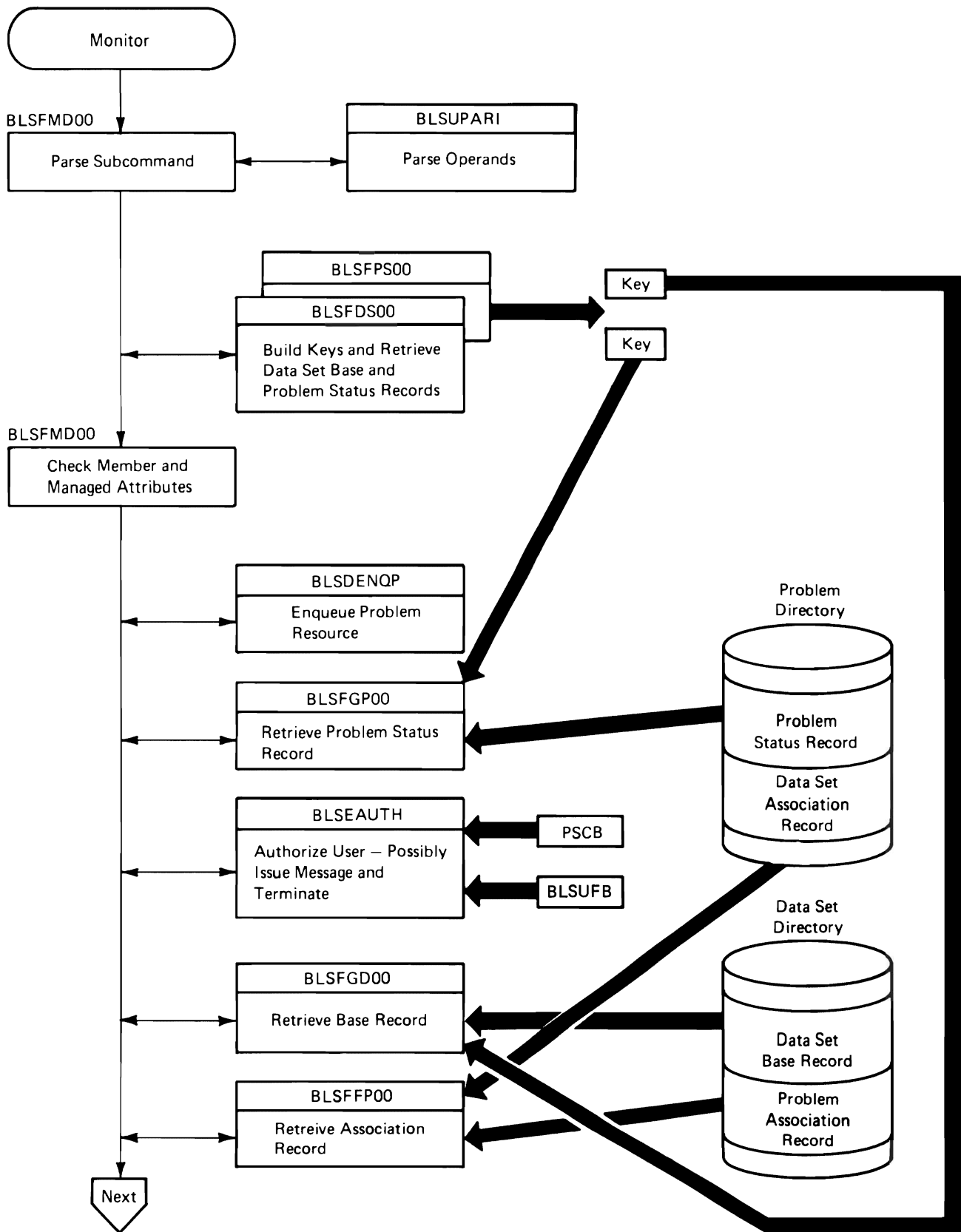


Figure 16 (Part 1 of 2). MODDSN Subcommand

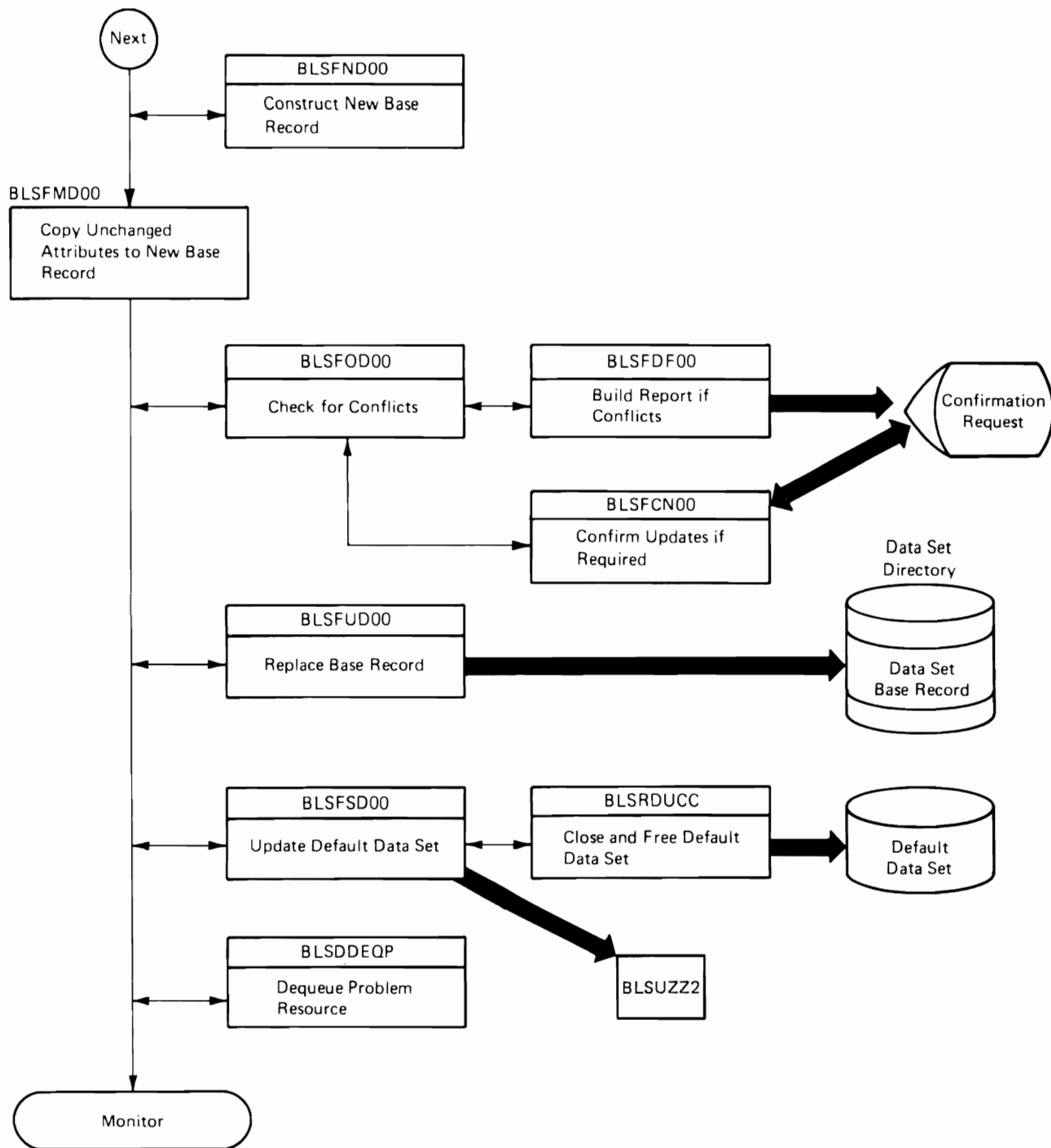


Figure 16 (Part 2 of 2). MODDSN Subcommand

Analysis Subcommands

The analysis subcommands perform checks and display dump information from the various components specified.

ASMCHECK Subcommand

BLSRASM7 is the module that processes the ASMCHECK subcommand.

BLSRASM7 accesses, compiles statistics, analyzes, and produces diagnostics for control blocks associated with the auxiliary storage manager component as shown in Figure 17 on page 56.

The following steps are done:

ASMVT fields are inspected: After parsing entered operands and setting routing indicators, the CVT address is obtained via a call to BLSRESGU. All dump access is done by BLSRACC. The ASMVT is obtained using the CVTASMVT field and then the ASMVT address is displayed, using BLSUMPK1 and BLSUPUTO. The ASMIORQR and ASMIORQC fields are converted to decimal and displayed.

The PART is examined: Next, the ASMPART field is used to get the PART. If the PARTIDEN field does not contain the characters 'PART', a message is produced and the program exited. The total PART length is calculated using the PARTSIZE field, and the whole PART with PART entries is obtained. An equate symbol record is created for the PART, and the PART address is displayed.

Each PART entry is examined: The PAREUCBP field is used to get the UCB for that page data set and the PARENEN and UCBNAME fields are used to produce a message displaying the device containing that numbered page data set.

Paging activity is displayed: Using the PAREIORN and PAREIORB, the IORBs for each PART entry are obtained, and the IORID field is validity checked. If the check fails, the program is exited. For each IORB, the IORIOSB field is used to obtain the IOSB. If the IORB is in use, a message indicating I/O is active is displayed.

If I/O was active, the IOSB return code is checked against IOSNRMC (which is not necessarily zero), and, if the two codes are not equal, the return code is displayed.

Upon completion of the inspection of each PART entry, the program terminates.

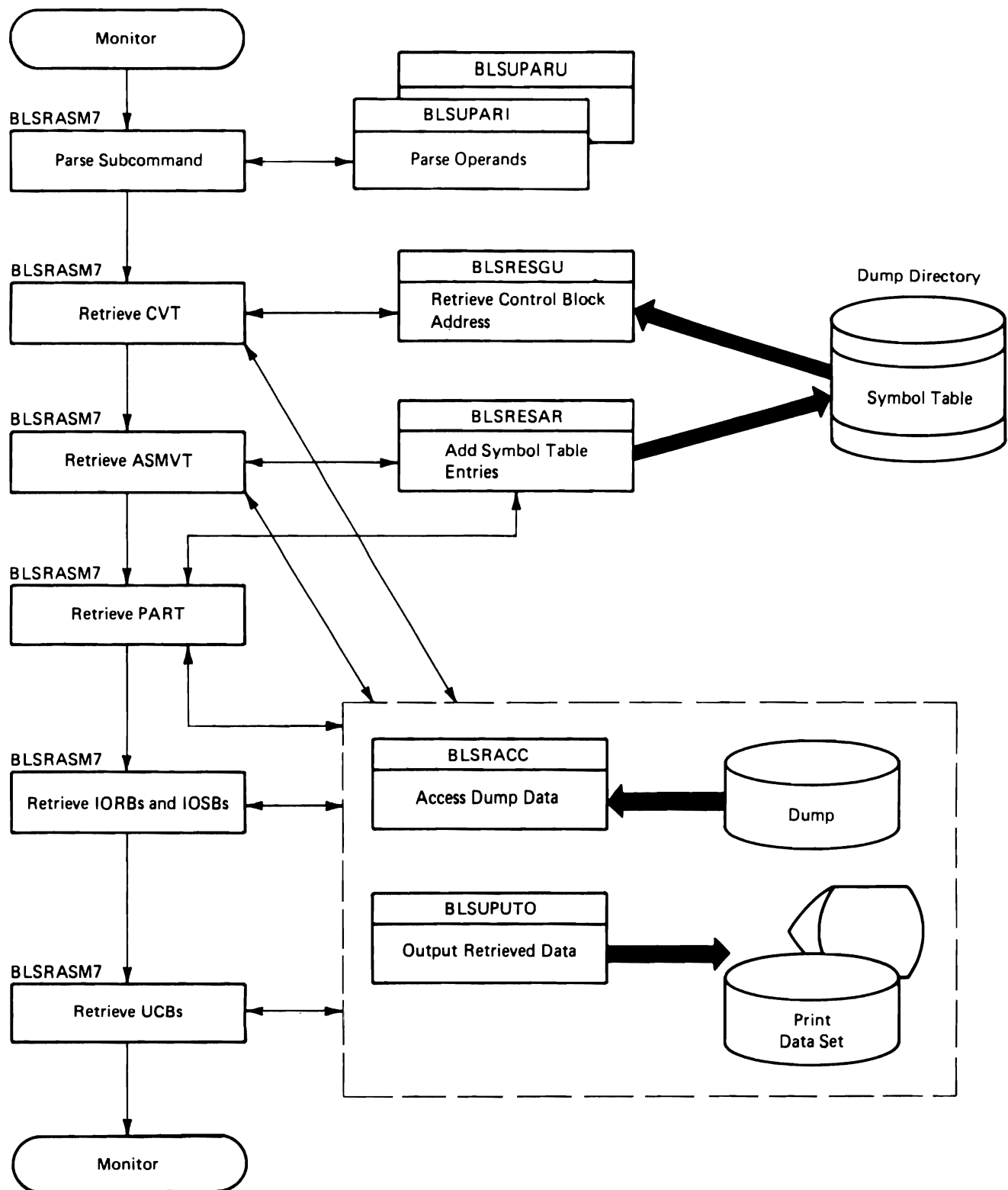


Figure 17. ASMCHECK Subcommand

COMCHECK Subcommand

BLSRCOMK is the module that processes the COMCHECK subcommand.

BLSRCOMK accesses, compiles statistics, analyzes, and produces diagnostics for control blocks associated with the communications task component (COMMTASK) as shown in Figure 18 on page 58.

The following steps are done:

The required UCM fields and pointers are obtained: After parsing entered operands and setting routing indicators, the UCM address is obtained via a call to BLSRESGU. Next, the UCM is obtained (all dump access is done by BLSRACC), and the WQE pointers, ORE pointer, UCME pointers, buffer limit, ORE and WQE counts, and COMMTASK's ASID are saved.

The user is informed of COMMTASK's activities: The WQE count and WQE limit are displayed in a message. The WQEs chained from the UCM are obtained and counted and a message is issued displaying the count if the chain is intact.

The status of each console is inspected: Each UCME is inspected. If the status byte is nonzero, a message is produced. Each CQE chain is processed and the WQEs counted. If any WQEs are found, a message is produced after obtaining the unit name from the UCB pointed to by the UCME.

Lastly, the OREs are inspected: The ID number and the text of the message are displayed.

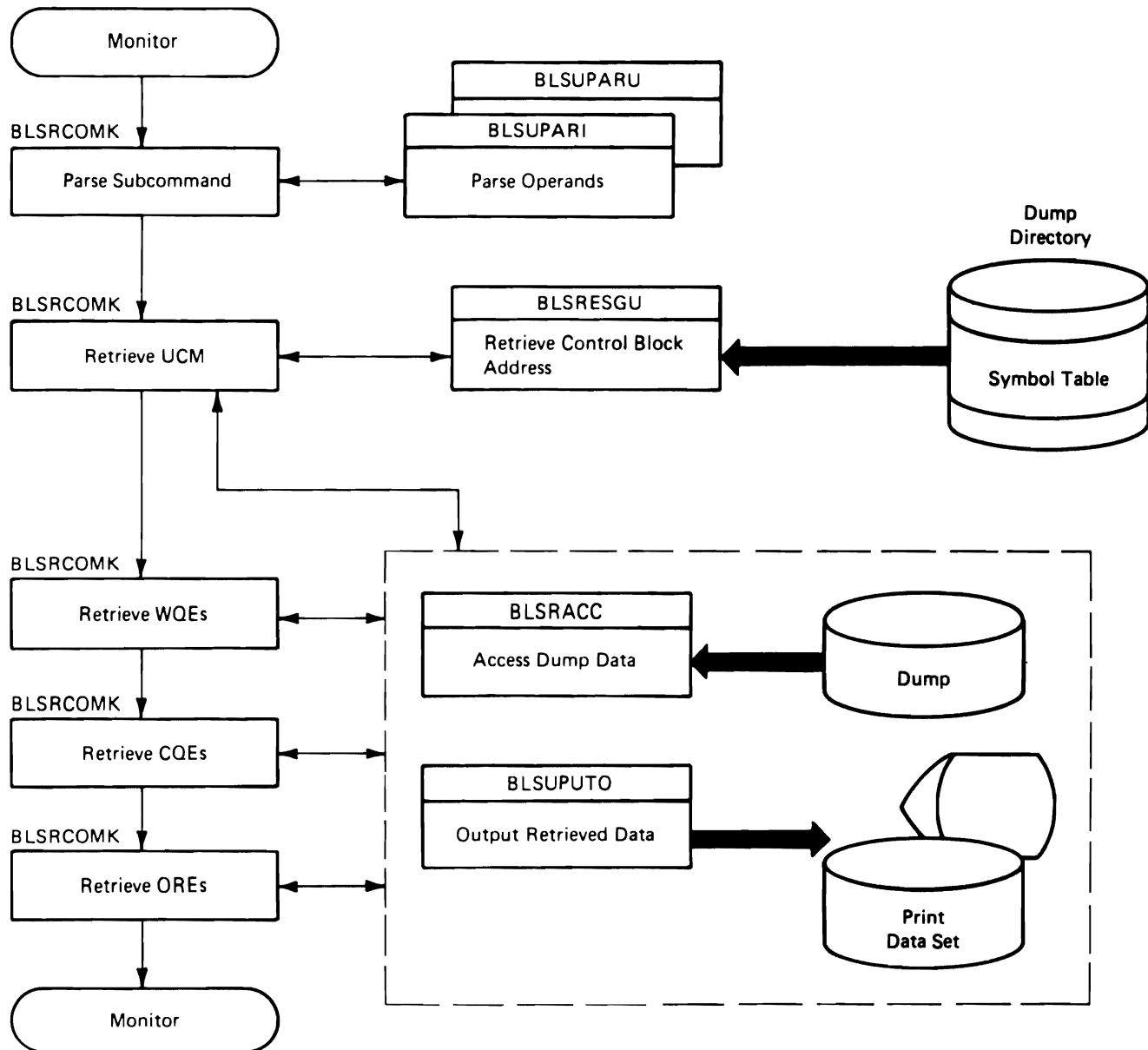


Figure 18. COMCHECK Subcommand

ENQCHECK Subcommand

The ENQCHECK subcommand displays the ENQ/DEQ resource status shown in the dump. The module that controls ENQCHECK processing is BLSRENQK. It is the entry point of load module BLSRENQK which also contains BLSRENPA (with its data CSECT BLSRENPC), BLSRENQA, and BLSRM077.

BLSRENPA first parses the subcommand and processes the operands as shown in Figure 19 on page 60.

BLSRM077 is called to warn the user that ENQCHECK processing may be invalid if a virtual dump is being processed. The processing may be invalid because the system was running while the dump was being taken.

BLSRENQA is called to perform the actual processing of ENQ/DEQ control blocks and to generate the requested report.

The ISGGVT is located in the dump (BLSRESGU). If global resource sharing was active (GVTGRSNA) in the dumped system, the ISGRSV is also located and both the global and local queue hash tables, ISGQHTG and ISGQHTL respectively, are accessed.

Each ISGQCB is validated (BLSRESSA) and retrieved from the dump (BLSRACC). An absolute maximum of 1000 ISGQCBs are processed so that no infinite loops may occur.

- If a major name was specified in the subcommand and the ISGQCB's major resource name doesn't match it, the ISGQCB is ignored and processing continues with the next ISGQCB.
- If the ALL option was omitted from the subcommand, the ISGQCB (QCBNOENQ) and enough associated ISGQELs (QELSHARE) are examined to determine whether the resource was subject to contention. If there was no contention for the resource, processing continues with the next ISGQCB.
- Otherwise, a message (BLS18041I) is issued which identifies the resource name, address, and the scope of the resource name. In the case of a step resource the affected ASID is included in the message. After the message is printed, the ISGQEL chain of the ISGQCB is processed.

Each ISGQEL is validated (BLSRESSA) and retrieved from the dump (BLSRACC). A message (BLS18042I) is displayed for each ISGQEL. This message identifies the address, owning ASID, and use (shared or exclusive) of the resource. For global resources the owning system is identified along with the owning ASID within that system. The name of the owning system is used if it can be accessed in the ISGRSV (RSVESYNM). Otherwise, the numeric system identifier (QELSYSID) is used.

If the resource is for a reserved device, the associated UCB is validated and retrieved from the dump, and a message (BLS18116I) is issued. The message displays the name and address of the UCB.

After all ISGQCBs, ISGQELs, and UCBs have been processed the subcommand terminates. A message (BLS18063I) is issued if a major resource was named in the subcommand but was not found.

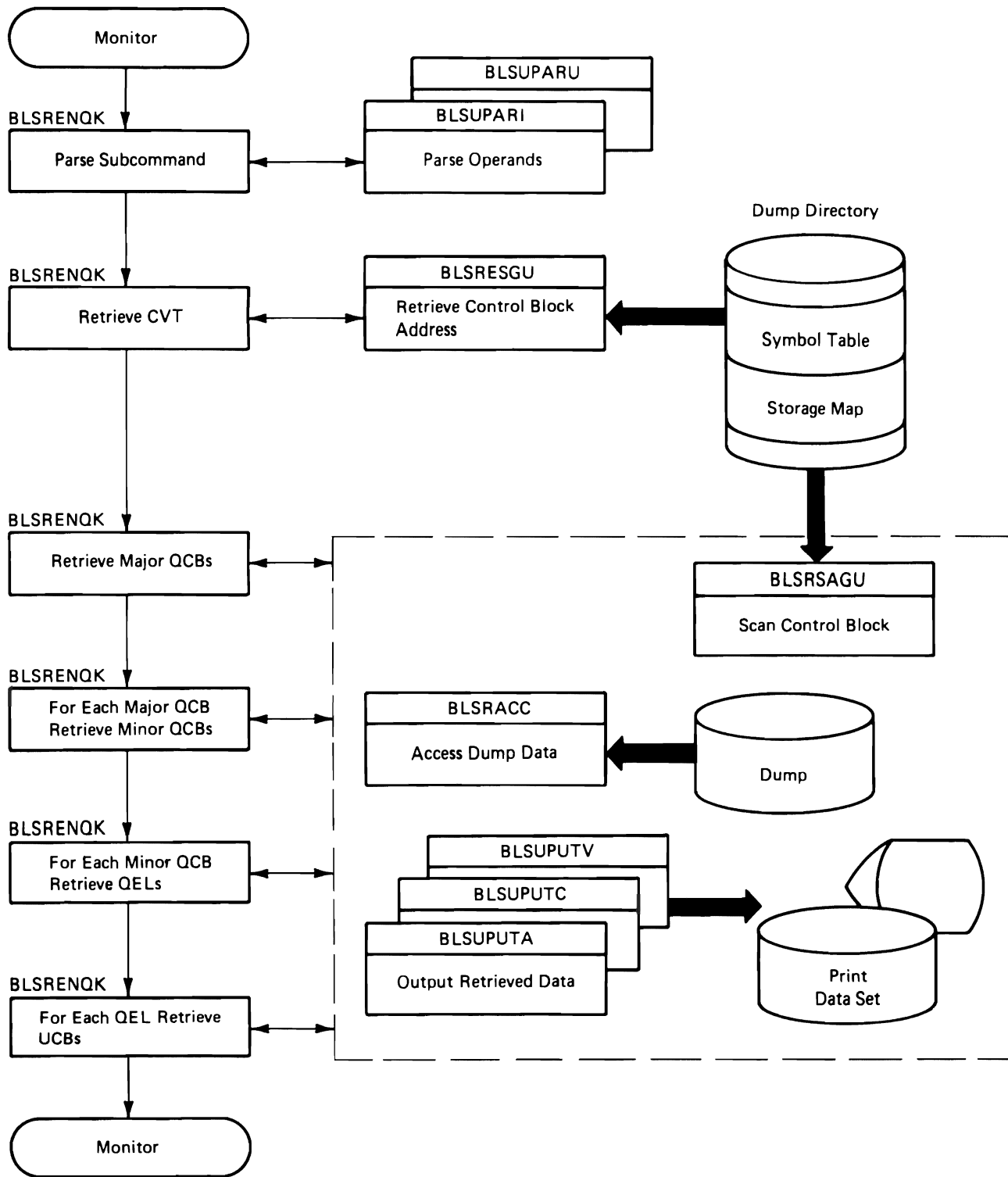


Figure 19. ENQCHECK Subcommand

STATUS Subcommand

BLSRST01 is the module that processes the STATUS subcommand, as shown in Figure 20 on page 62.

If system status is requested, BLSRST01 accesses and formats the nucleus identifier from the OS/VS1-OS/VS2 common CVT extension (CVTXTNT2). BLSRST01 also accesses the time-of-day clock value from the header record of the dump and the software adjustment factor (CVTTZ) from the CVT, formatting the dumped time-of-day clock followed by an adjusted value.

If CPU status is requested, BLSRST01 accesses and formats the PSW from the header record of the dump. If requested, BLSRST01 also accesses and formats general purpose and control registers. For SADMP the vector registers are formatted for each processor that has the Vector Facility installed. The current ASCB and TCB are validated and their addresses are displayed. This information may be displayed for several processors in a multiprocessing configuration.

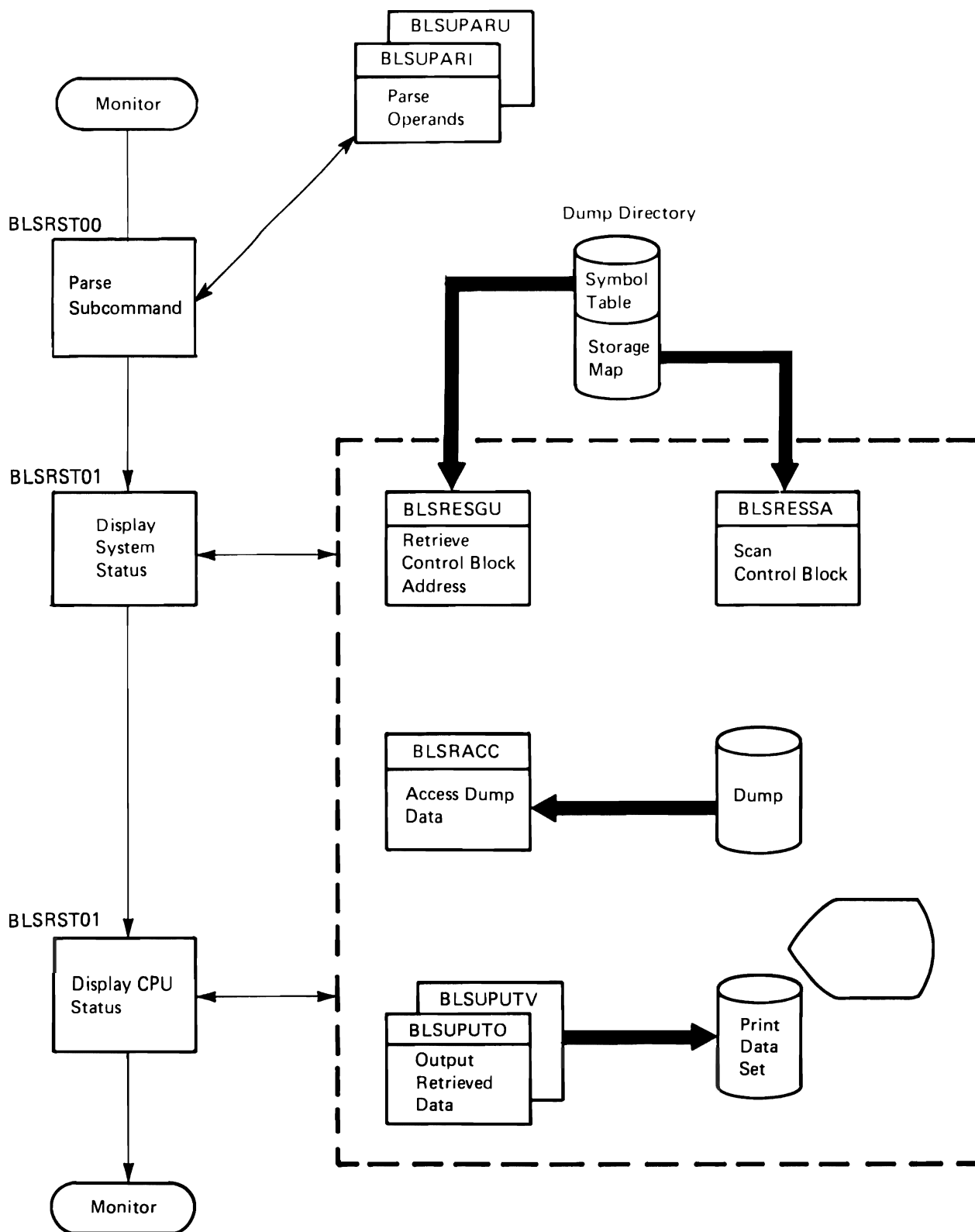


Figure 20. STATUS Subcommand

SUMMARY Subcommand

BLSRSUMM is the module that processes the SUMMARY subcommand.

BLSRSUMM displays selected fields from all or selected ASCBs and associated TCBs, RBs, and CDEs. BLSRSUMM also creates symbol table entries for each major control block encountered during processing and scans certain control blocks.

The SUMMARY subcommand has format control keywords and selection keywords that provide a variety of output formats and a selectivity of address spaces to be processed. BLSRSUMM creates one of four report/displays as specified by the format control keywords KEYFIELD, TCBSUMMARY, JOBSUMMARY, and FORMAT and processes address spaces as specified by the selection keywords ALL, CURRENT, ERROR, TCBERROR, ASIDLIST, and JOBLIST.

The following steps are performed:

The exit environment is established

The input character string is parsed: BLSRSUMM links to IKJPARS to validity check the format control and selection keywords that the user entered. BLSRSUMM places the results in the BLSQSMPL, which is a data area used to communicate among the several summary modules. (See "BLSQSMPL" on page 361.) BLSRSUMM also sets the routing indicators.

The SUMMARY subcommand differs from other subcommands in that no output is produced when the NOTERMINAL and NOPRINT options are in effect. This allows the user to employ SUMMARY solely to generate symbols.

The SUMMARY exit program BLSQSUM1 is called: Defaults are established for omitted keywords. The select ASID service is called via the exit service router (BLSQROUT) to create a list of ASCB pointers that meet the selection criteria of the specified or default selection keywords. The address of the select service output is placed in BLSQSMPL. If no address spaces are selected, a message is produced indicating this condition.

The CVT is accessed: If either the JOBSUMMARY or the FORMAT keyword are specified, BLSRSUM1 accesses the CVT from the dump. Pointers to various areas and control block chains that are to be processed by the report processing modules are extracted and placed in BLSQSMPL.

Report processing modules are called: One or more of the following programs are called in response to the specified or default keywords:

KEYWORD	MODULE
FORMAT	BLSQSUM2
KEYFIELD	BLSQSUM3
JOBSUMMARY	BLSQSUM4
TCBSUMMARY	BLSQSUM5

Terminal processing: If the select ASID service produced any output, BLSRSUM1 releases storage obtained for processing and BLSRSUMM releases the parsed PDEs.

The report processing modules call the following common exit services via BLSQROUT:

- The storage access service obtains an image of the ASCB and processes the specified format control keywords.
- If the FORMAT keyword is specified, the control block formatter formats all the control blocks in their entirety.
- The format service processes all the format keywords except FORMAT. This service uses a format pattern containing only the fields in the control block that are defined for the specified display/report. TCB exits are invoked to format the I/O control blocks.

If the KEYFIELD and REGISTERS keywords are specified, the register contents saved in the TCBs and the RBs are formatted and printed. Otherwise the registers are not printed.

- The equate symbol service adds a symbol to the symbol table for each ASCB accessed.
- The print service prints all reports.
- The index service adds index entries.

Final processing: The storage for the ASID list is freed.

PRDMP Exit Support Subcommands

There are four subcommands that execute PRDMP's exits within the IPCS environment. They are ASCBEXIT, IOSCHECK, TCBEXIT, and VERBEXIT. The subcommands are processed by the modules BLSRASCX (ASCBEXIT), BLSRIOSK (IOSCHECK), BLSRTCBX (TCBEXIT), and BLSRVRBX (VERBEXIT).

After parsing the subcommand line (IKJPARS), all the subcommands complete the exit parameter list (BLSABDPL). The parameter list contains pointers to the print routine (BLSUPUTT or BLSUPUTI), the storage access routine (BLSRACCL or BLSRACCI), and the PRDMP format routine (AMDPRFMT). These routines simulate the PRDMP environment within IPCS. BLSABDPL is imbedded at the beginning of the IPCS task variable (see "BLSUZZ2" on page 406) so that the simulation routines (which are reentrant) can execute in the IPCS environment when they are passed pertinent information within the BLSABDPL by the exit.

All the subcommands fill in the address of the CVT (ZZ2AMDC).

The ASCBEXIT subcommand fills in the ASID (ZZ2AMDA).

The IOSCHECK subcommand fills in the parameter list extension (ZZ2AMDXC). If an enquoted string is specified in the subcommand or the default string, 'EXCEPTION', is accepted, the parameter list extension points to the string (ZZ2AXOT); if a null string is designated, the parameter list extension is zero.

The TCBEXIT subcommand resolves the address specified in the subcommand line and stores the address and ASID in the TCB address field (ZZ2AMDT). No other

data descriptions of an IPCS address (such as LENGTH) are passed in a PRDMP exit.

The VERBEXIT subcommand fills in the parameter list extension (ZZ2AMDXC). If an enquoted string is specified in the subcommand, the parameter list extension points to the string (ZZ2AXOT); if not, the parameter list extension is zero.

The subcommands all invoke the exit routine via a LINK SVC. A diagnostic message is issued if the LINK fails. The return code from the exit is used as the return code from the subcommand. BLSQECT checks the auxiliary return code in the parameter list extension and prints a message, if appropriate. The auxiliary return code does not affect the return code from the subcommand.

Figure 21 summarizes the various pointer fields within the BLSUZZ2 control block used by these subcommands.

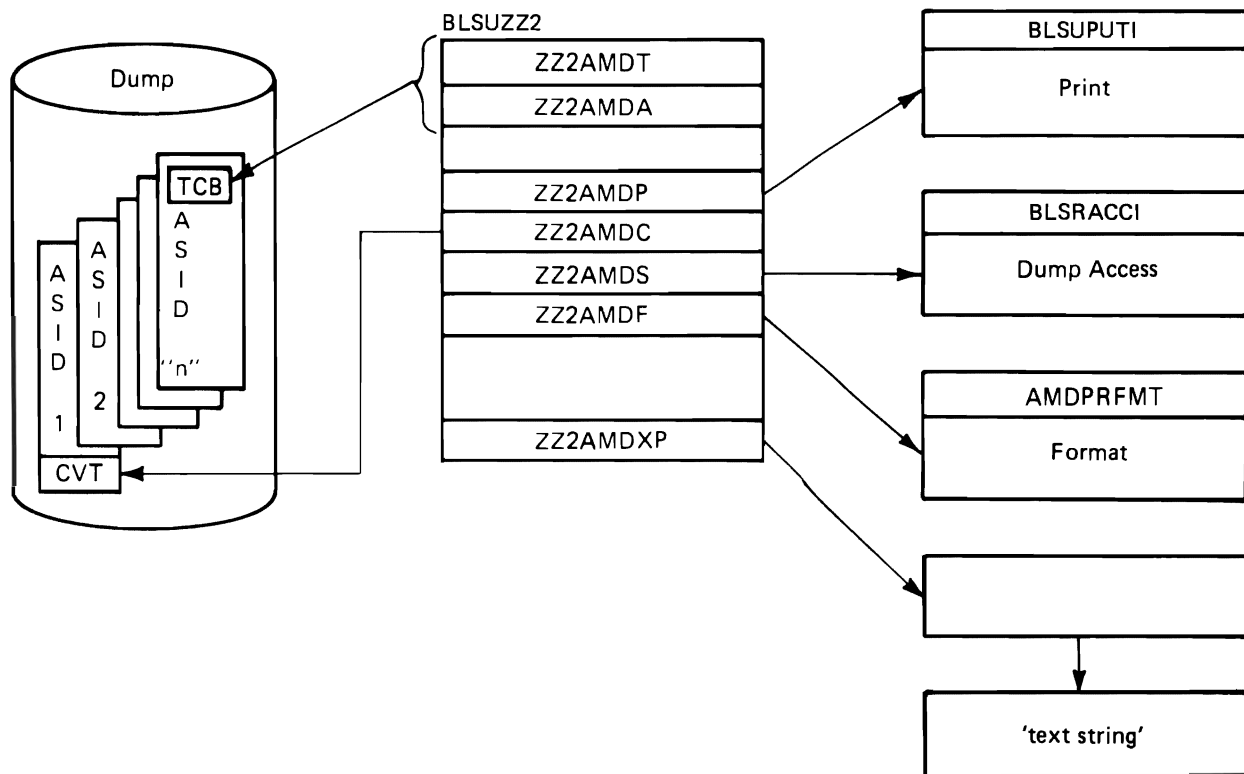


Figure 21. Simulated PRDMP Parameter List

Line-Oriented Dump Viewing Processing

The line-oriented dump viewing subcommands are used to find and display various portions of the dump and run through chains of control blocks.

FIND Subcommand

BLSRFIND searches a given area of the dump for a specified search argument (such as a module name or any hexadecimal or character value), and optionally displays the data.

The processing for this subcommand occurs in three phases:

- Setting up search parameters
- Performing the search
- Displaying results

Setting Up Search Parameters: If a search argument is entered, it is translated into binary form (if necessary) by BLSRVAL. If an address is not entered, the beginning address is set to X+1.

Next, the beginning search address is adjusted for boundary and offset alignment.

If no address is entered, and the computed beginning address is outside of FINDAREA, a message is issued and the subcommand terminates.

The resultant operands from the above processing are now saved for future finds. Note that the mask value is not saved, nor the BREAK/NOBREAK indicator.

The updated X and FINDAREA symbols are placed in the symbol table by BLSRESAR.

If a mask is entered, the mask is validity checked and converted to binary. Note that the mask must match the search argument length with a maximum of eight bytes. The mask is then ANDed with the search argument.

If no search argument is entered, the previously entered FIND subcommand's boundary, offset, and search arguments are used. After parsing entered operands and setting routing and display indicators, the symbols X and FINDAREA are retrieved from the symbol table by BLSRESGE. If an address operand was entered, BLSRADDs is called to resolve it. If not, the search range is set as beginning at X+1, and ending at the end of FINDAREA, or, if the FINDAREA symbol does not exist, at location X'7FFFFFFF'.

If no search argument is entered with the subcommand, and this is the first time FIND is entered, a message is issued, and FIND subcommand processing terminates.

Performing The Search: Service routine BLSRFISE performs the search.

Displaying Results: If the data was found, BLSRESAR updates symbol X to the data address. If the VERIFY operand is specified, BLST01 is called to display the data, and the subcommand terminates.

If the loop processing completes without a found condition, a message is issued, and processing terminates (return code 4).

FINDMOD Subcommand

BLSRFMOD is the module that processes the FINDMOD subcommand.

BLSRFMOD searches the symbol table, link pack area CDE queue, and link pack area directory entries for a given module name.

After parsing entered operands and setting routing, display, and verification indicators, the module name is prepared for use by either saving it as entered, if the character keyword was entered or implied, or converting it to a bit string. If hexadecimal characters were entered, validity checking is done for length and valid characters, and a message displayed. If the validity checks fail, the program is exited.

Next, an equate symbol record image is filled in with the requested name, and a call to BLSRESGE is made to see if the symbol is in the table. If the symbol is found and the type field indicates module, the module is considered found. If verification is requested, BLST01 is called, and the program is exited.

If the name is not in the symbol table, the CVT address is obtained via BLSRESGU, the CVTQLPAQ field is obtained (all data access is done by BLSRACC), and the CDE queue is accessed, entry by entry until a match is found or the end of the queue is reached. If a match is found, a symbol is created (all symbols are created via BLSRESAR) for the name CDEPROGNAME. If it was a minor CDE, the major CDE is obtained, its name displayed and it is added to the table. The extent list is obtained, symbol XLPROGNAME is added to the table, and the module name itself is added. If verification is in effect, BLST01 is called to display the module data and the program is exited.

If the name was not in the CDEs, the LPDE hashing algorithm involving the CVTVVMDI and CVTDIRST fields from the CVT, and an exclusive ORing of the two halves of the module name, is used to get an LPDE chain address. The chain is searched for a matched name. If not found, a message is issued and the program is terminated. If found, and it is a minor LPDE, symbol LPDEPROGNAME is added to the symbol table, the major name is displayed, and the LPDE search algorithm is used to get the major LPDE. Name LPDEMAJORNAME is added to the table, and the module name itself is added to the table. If verification is in effect, BLST01 is called to display the module data and the program is exited.

FINDUCB Subcommand

BLSRFUCB is the module that processes the FINDUCB subcommand.

BLSRFUCB parses the operands, and calls the service routine that finds a requested UCB, then optionally displays it.

After parsing entered operands and setting display options, an equate symbol record is filled in with the unit address requested. BLSRESGU is called to initiate a search for the requested UCB. If the search is successful, the equate symbol for X is updated to the UCB address, and, if verification is in effect, BLST01 is called to display the UCB.

LIST Subcommand

BLSRLIST is the module that processes the LIST subcommand.

BLSRLIST displays a requested amount of dump data. After parsing entered operands and setting the display options, a loop is entered to process each of the address expressions entered. The list and range options for the IKJPOSIT address PDE are used, so each list item is processed separately. Note that if a list of address expressions was entered, they are enclosed in parentheses and any modifying operands (ASID, length, etc.) entered outside the parentheses apply to all the items in the list.

Each address expression except the last (or only) is resolved via a call to BLSRADDR; the last expression is resolved via BLSRADD, which also updates X. For each resolution that returns a code less than 12, BLST01 is called to display the data.

RUNCHAIN Subcommand

BLSRRNCH is the module that processes the RUNCHAIN subcommand.

BLSRRNCH accesses, counts, and optionally names and displays a specified chain of control blocks. The user specifies (or allows to default) the starting address, length, offset of forward pointer, maximum number of blocks to process, end-chain pointer value, forward pointer mask, and a name prefix to create symbol table entries.

After parsing entered operands and setting routing and display indicators, BLSRRNCH resolves the starting address. The minimum number of digits needed to suffix the name prefix is calculated to satisfy the maximum-number-of-blocks parameter. If the resultant name is greater than 31 characters, the user is informed, and the subcommand terminates.

BLSRRNCH processes the requested chain of control blocks as follows:

1. Calls BLSRACC for each access of dump data. The high-order byte of each forward pointer is set to zero.
2. Obtains each control block, using the forward pointer field of the previous block, until either the maximum specified number have been processed, or the forward pointer, when ANDed with the mask supplied, is equal to the specified end-of-chain indicator. If necessary, the length is adjusted to include enough of the block to encompass the forward pointer field.
3. If requested, adds a symbol table entry (equate symbol record) using the supplied prefix and a suffix beginning with 1 (or 01, or 001, etc.) for each block. BLSRESSA and BLSRESSR are called to process the symbol.
4. If the control block is one that IPCS can scan, calls BLSRESSR to process the symbol. For the first block, scan exit defaults might be placed into the block description.
5. If verification was requested, calls BLST01 to display the data.
6. If the EXEC option was requested, creates a special environment control table (ECT) using the TSO macro IKJECT during BLSRRNCH initialization. This

table is discarded after BLSRRNCH terminates. BLSRRNCH calls four subroutines to process the EXEC option for each block processed:

- BLSRESAR updates symbol X to describe the current block.
- BLSUSTKS stacks the subcommand or CLIST invocation, using the special environment control table to separate BLSRRNCH command processing data from the data used by the caller of BLSRRNCH.
- BLSUMON processes the subcommand or CLIST.
- BLSRDUSO reestablishes access, if necessary, to the data source specified on the RUNCHAIN subcommand.

7. At the end of RUNCHAIN's processing, notifies the user of the number of control blocks that were processed.

Full-Screen Dump Viewing Processing (DSPL3270)

The DSPL3270 subcommand provides a full-screen dump viewing capability which utilizes an expanded set of features available on a 3270, such as program function (PF) key definition, tabbing, and cursor positioning. A formatted screen image is generated which permits the entry of data in predefined fields and allows the user to specify and control the format and content of dump data to be displayed upon subsequent screen image generation. An interactive environment is established during DSPL3270 initialization, in which terminal input/output as well as attention key acknowledgment is performed under the control of the DSPL3270 subcommand. Most DSPL3270 processing is performed by CSECT BLSR327A, as shown in the following diagrams. The interactive environment, as shown in Figure 22 on page 72 permits the following logical sequence to be maintained:

1. Establish full-screen mode and generate an initial screen image.
2. Await user response.
3. Perform response processing.
4. Generate a modified screen image.
5. Repeat cycle from 2.

DSPL3270 performs the processing necessary to terminate this cycle and exit full-screen mode at the user's direction.

Full-Screen Initialization

Initialization processing, as shown in Figure 23 on page 73, consists of three steps:

1. Checking subcommand operands - The IPCS service routine BLSUPARI is invoked as an interface to the TSO parsing routines to establish the validity and values of the operands entered with the subcommand. If the PAUSE operand was supplied, a message is transmitted to the terminal instructing the user to notify DSPL3270, by depressing the ENTER key, when full-screen mode initialization should resume. No further processing is performed until this response is received.
2. Establishing initial values - On entry, DSPL3270 automatically generates a default screen image. The content and format of this screen image is governed by a keyword operand (RESET/NORESET), in conjunction with any existing dump directory information describing the full-screen image established for the

current dump. Service routine BLSRPHGE is invoked to obtain the existing information. Additionally, symbol management modules BLSRESGU and BLSRESGE are called to obtain the storage images of, respectively, the system communications vector table (CVT) and the IPCS current symbol (X), each of which may be used to generate the initial screen image.

3. Establishing full-screen mode - The TSO terminal input/output control (TIOC) program must be notified that an application wishes to assume full-screen control:
 - Installations using TCAM/TIOC implementations must incorporate special modifications when their message control program (MCP) is assembled. DSPL3270 identifies each full-screen message to the modified MCP through the use of a harmless nonsense prefix to its graphic orders (X'115D7F11').
 - VTIOC with VTAM provides integrated support for full-screen mode. DSPL3270 uses SVC 94 (macro STFSMODE) to enter full-screen mode during DSPL3270 initialization. The return code from SVC 94 is used to determine the presence or absence of VTAM.

Screen Image Generation

Screen image generation consists of constructing an output buffer containing the 3270 graphic orders and data necessary to produce the desired screen image and transmitting this buffer, via a TPUT macro, to the terminal. This is shown in Figure 24 on page 74. The requisite orders and data are determined by merging a knowledge of the current screen image with a definition of the desired image.

DSPL3270 maintains a work area which describes the content and format of the current screen image. Initially, this area contains the values established during step 2 in the preceding section, and also describes the desired screen image. During interactive processing the user may modify portions of the screen or issue requests which result in DSPL3270 altering the screen. These modifications and requests are recognized and reflected in the work area during response processing.

The output buffer construction discipline calls for the transmission of a minimum amount of graphic orders and data. Thus, only those orders and data necessary to display new or altered portions of the screen are transmitted. Dump data to be displayed is obtained by calling BLSRACCL.

Response Processing

A TGET macro is issued immediately following the TPUT, and DSPL3270 enters a wait state which is terminated by the user signaling a response via an interrupt. Response processing is shown in Figure 25 on page 75, and is performed in the sequence described below:

1. The source (light pen, PA key, PF key, ENTER key) of the interrupt is determined. This determination influences which, if any, of the following steps are performed. For example, depressing a PA key results only in an interrupt; no input data is received. Hence the screen is regenerated immediately. Similarly, within DSPL3270, PF3 is defined to signal a request to terminate subcommand processing, which is initiated immediately.

2. Cursor positioning is noted and saved for later use. As an exception, positioning under the field which signifies subcommand termination is honored immediately.
3. The input data (if any) is processed sequentially. Input field modifications are noted and checked for validity (BLSRVPAS is invoked to perform ASID validity checks). If erroneous input fields are detected a new screen image is generated in which erroneous fields are intensified. Response processing begins anew on the subsequent interrupt. Modifications to portions of the screen not defined by DSPL3270 as input fields are recognized and interpreted to represent inadvertent screen image destruction (the previous screen image is regenerated).
4. If the cursor was positioned at a significant field on the screen, the appropriate processing is performed (for example, under a word in the dump display area, which represents an indirect addressing request).
5. If a predefined PF key was pressed, the processing required to accomplish the requested function (for example, scrolling) is performed.
6. Work area values that reflect screen image modifications and define the desired screen image are updated.
7. If the subcommand field was modified, its contents are added to the TSO I/O service routine input stack.

Termination

Subcommand termination, as shown in Figure 26 on page 76 , is performed in two steps:

1. Exiting full-screen mode - The TIOC program must be notified when an application wishes to relinquish full-screen control:
 - TCAM/TIOC MCP modifications are informed through the use of a harmless nonsense prefix to the final set of graphic orders (X'115D7E11').
 - VTIOC with VTAM is informed through the use of SVC 94 (macro STLINENO).

Under either access method, lines 6-24 of the screen are cleared and the current line number is set to 6.

2. Establishing final values - On exit, DSPL3270 resets the IPCS current symbol (X) to describe the storage area last referenced within DSPL3270. This is accomplished by calling BLSRESAR. Additionally, the contents of the work area, which define the final screen image, are saved in the dump directory through a call to BLSRPHAR.

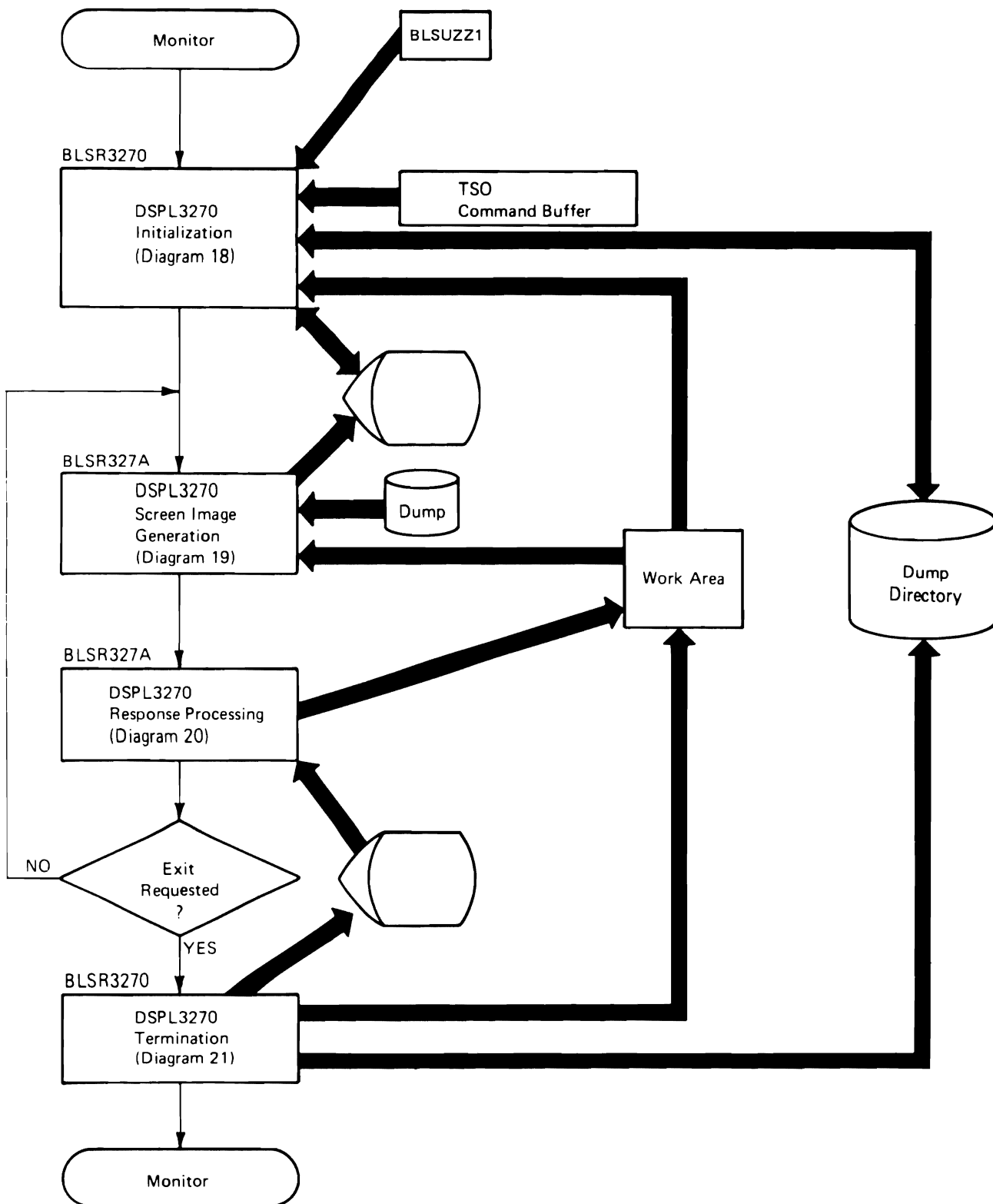


Figure 22. DSPL3270 Subcommand (Overview)

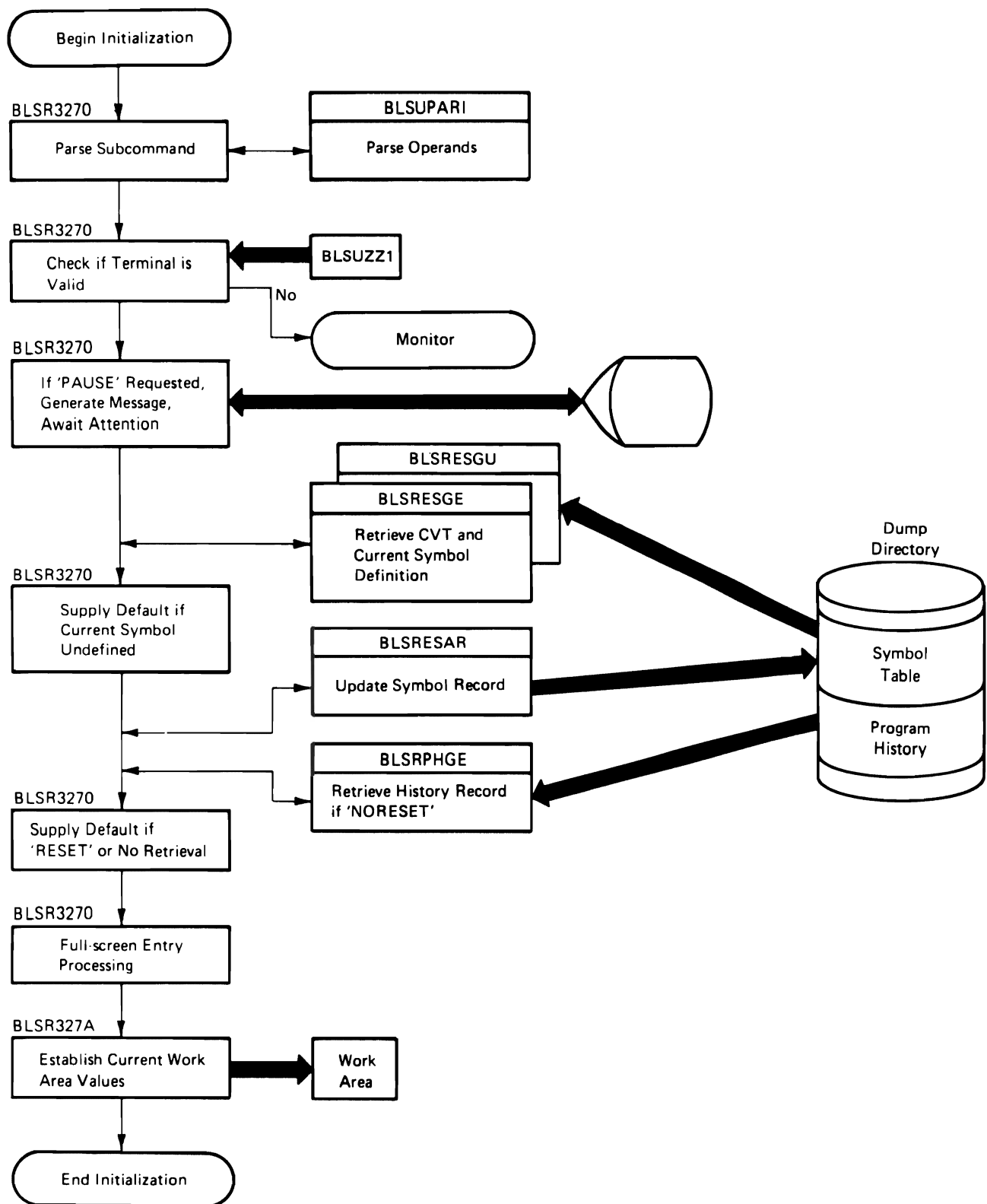


Figure 23. DSPL3270 Subcommand (Initialization)

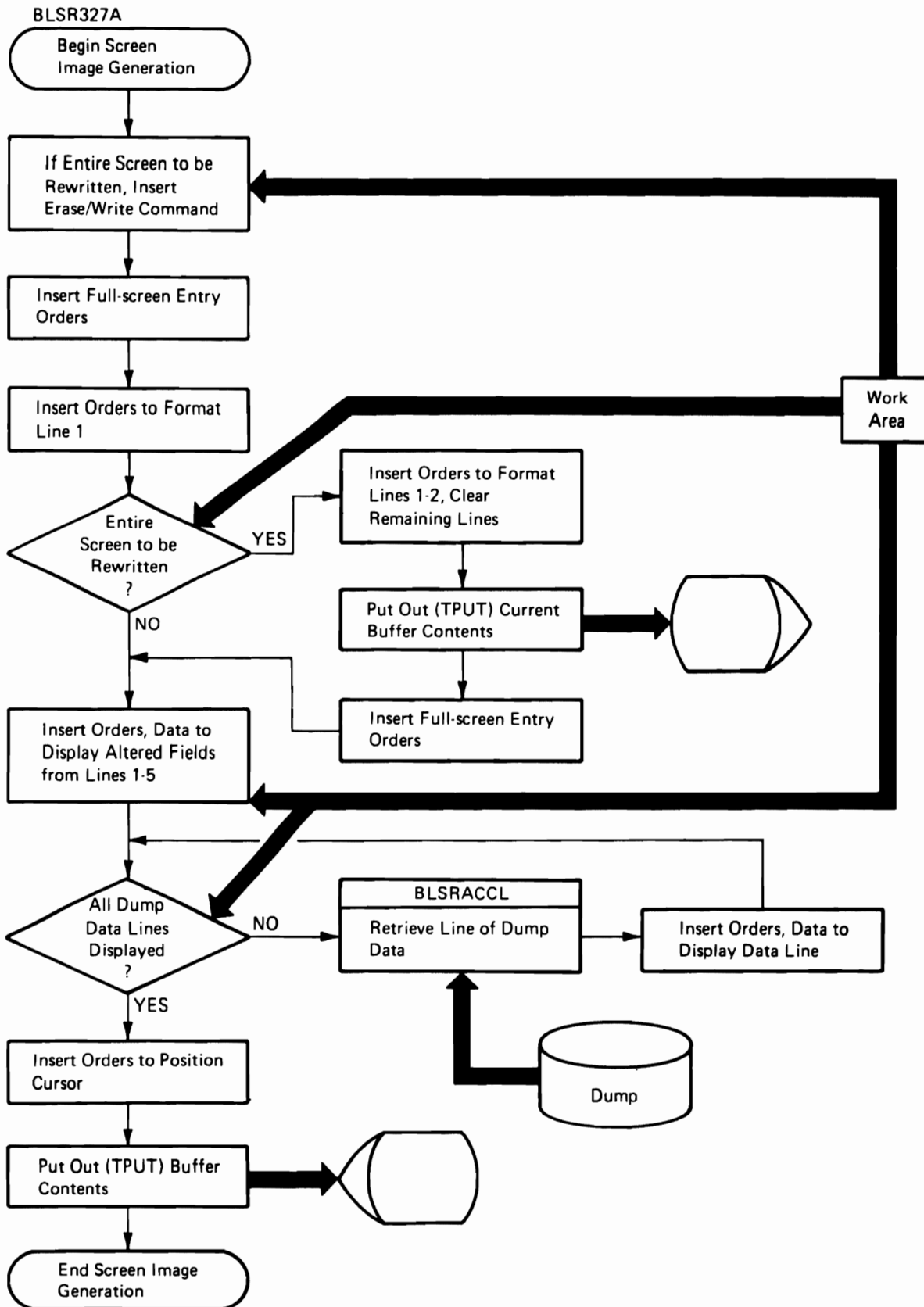


Figure 24. DSPL3270 Subcommand (Screen Generation)

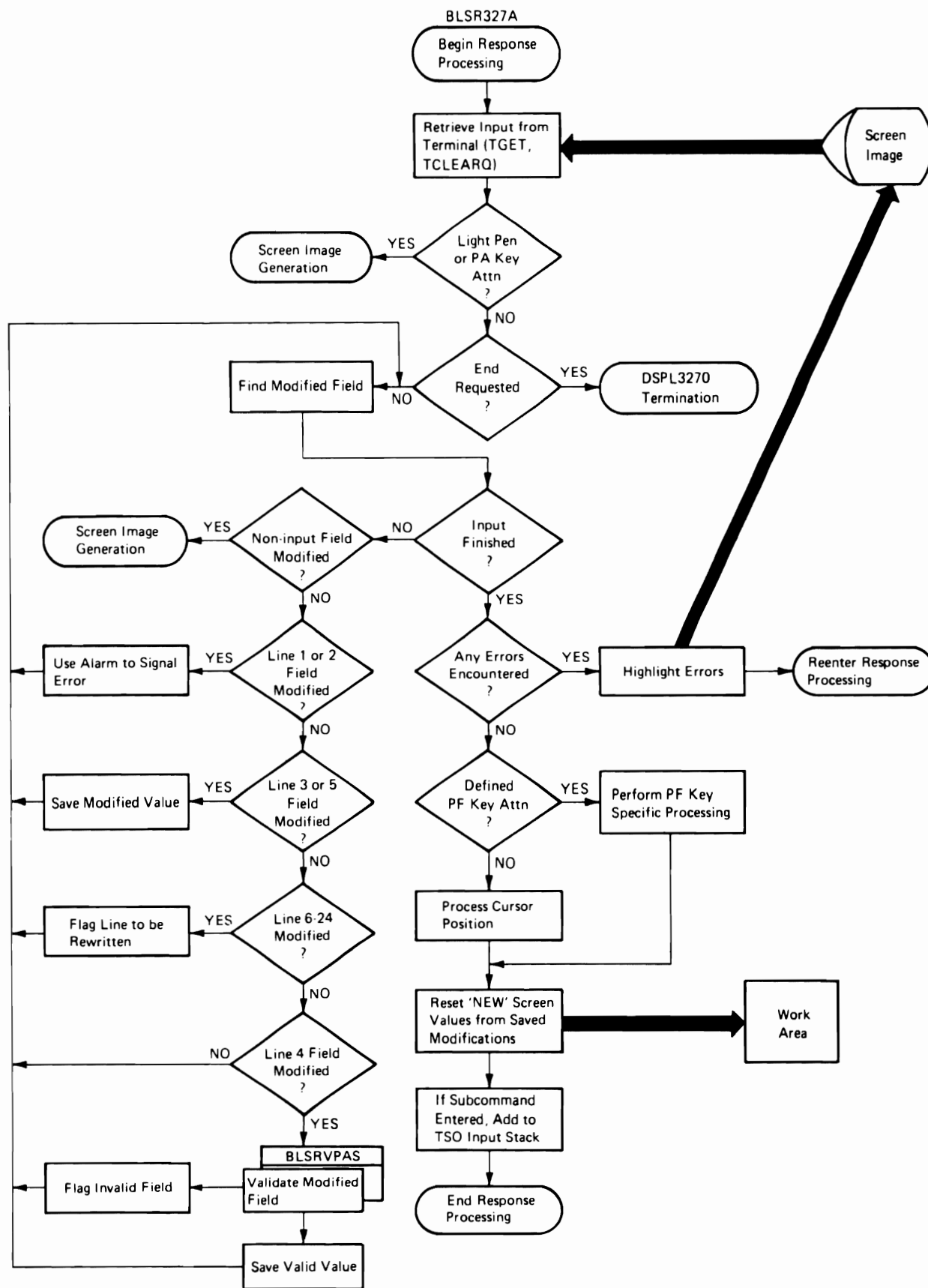


Figure 25. DSPL3270 Subcommand (Response Processing)

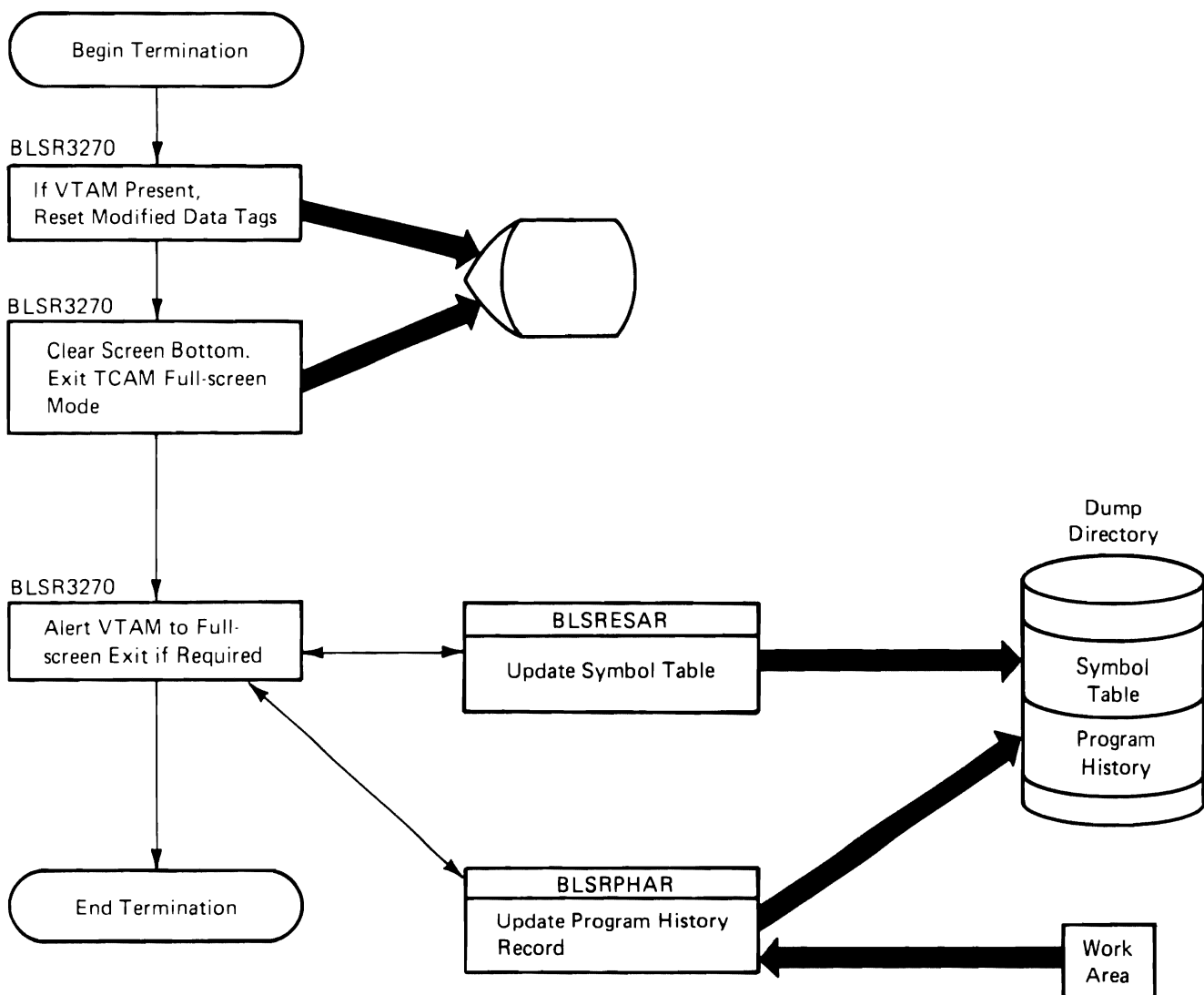


Figure 26. DSPL3270 Subcommand (Termination)

CLIST Support Subcommands

The CLIST support subcommands enhance CLIST execution control flow, and provide message production and pagination control for the print data set.

COMPARE Subcommand

BLSRCOMP is the module that processes the COMPARE subcommand.

BLSRCOMP does a comparison between values, a value and an area of storage, or two areas of storage. A mask value may be ANDed to the operands to ignore insignificant bits. Picture and text strings can also be used to check for patterns.

After parsing the operands and setting routing and flag indicators, the two operands are processed. If either one is an address, BLSRADDR is called to resolve it. If either one is a hexadecimal value, it is converted from character to bit. Character values are saved.

Two buffers are initialized, either by moving in the specified value, or accessing up to 4K of the dump data specified by the address for that operand.

The mask operand, if entered, is checked to ensure proper size and type, and is converted to a bit string for ANDing with the operands. A maximum of 16 hex digits is allowed, with no padding.

If the operand lengths are unequal, the user is informed via BLSUPUTO, and the compare length is set to the shorter of the two.

A loop comparing up to 4K bytes at a time is then begun. If requested, the mask is ANDed to the operands before the comparison is done.

If storage is not available before the compare length is depleted, the user is informed via BLSUPUTO, and return code 12 is set.

If the user specified the list option, BLSUPUTO is used to transmit a message indicating the result of the comparison.

EVALUATE Subcommand

BLSREVAL accesses one to four bytes of dump data, right-justifies them in register 15 and zeroes the high-order byte of the register.

After parsing the operands, setting the symbol X, and resolving the address, BLSRACC obtains up to four bytes of dump data, which are right-justified in a previously zeroed word. If not all the data was obtained, return is made with return code 12. (Note that the user cannot tell if this is a return code or data.)

The high-order byte of the word containing the data is then zeroed. This is because CLIST support does not return the high-order byte when accessing &LASTCC from a CLIST. The result is then placed in register 15, and return is made.

EVALDEF Subcommand

BLSREDEF is the module that processes the EVALDEF subcommand.

EVALDEF makes IPCS default data available to a CLIST of ISPF dialog via designated variables.

Numeric information may be formatted using either decimal or hexadecimal digits. Default formatting for numeric data is decimal.

EVALDUMP Subcommand

BLSREDUM is the module that processes the EVALDUMP subcommand.

EVALDUMP retrieves information associated with one data set described by the current dump directory and formats that information in CLIST of ISPF dialog variables.

EVALMAP Subcommand

BLSREMAP is the module that processes the EVALMAP subcommand.

EVALMAP retrieves information associated with one IPCS storage map entry and formats that information in CLIST or ISPF dialog variables.

Numeric information may be formatted using either decimal or hexadecimal digits. Default formatting for pointers and data used in conjunction with pointers is hexadecimal. Default formatting for other numeric data is decimal.

EVALSYM Subcommand

BLSRESYM is the module that processes the EVALSYM subcommand.

EVALSYM retrieves information associated with one IPCS symbol and formats that information in CLIST or ISPF dialog variables.

Numeric information may be formatted using either decimal or hexadecimal digits. Default formatting for pointers and data used in conjunction with pointers is hexadecimal. Default formatting for other numeric data is decimal.

INTEGER Subcommand

BLSULIN is the module that processes the INTEGER subcommand.

The integer contained in the first operand of the INTEGER subcommand is converted to displayable form for printing or terminal display. The output forms of OFFSET, POINTER, SIGNED and UNSIGNED are available. Also, either a CLIST variable or an ISPF dialog variable may be set to the value of the converted output.

For performance improvement, the INTEGER subcommand should be used in place of the HEX CLIST.

ISPEXEC Subcommand

BLSGX is the module that processes the ISPEXEC subcommand.

ISPEXEC allows a CLIST running as an ISPF dialog in an IPCS environment to initiate ISPF services. ISPEXEC might also be entered as an IPCS command on the command line of any of the full screen dialog panels.

BLSGX first determines whether ISP service routines, ISPEXEC and ISPLINK, are available. Then BLSGX passes the subcommand operands to ISPEXEC for processing by ISPF.

NOTE Subcommand

The NOTE subcommand accepts user-supplied text as input and transmits this text to the terminal, the printer, or both, as indicated by the default or specified routing. The processing is performed by BLSUNOTE.

Initial processing consists of parsing the operands entered with the subcommand (BLSUPARI). If the text is assigned a FLAG value lower than the message severity level currently in effect, processing is terminated. Otherwise, the PAGE keyword, if present, is honored by forcing a print page eject through a call to BLSUPUTA. Then, spacing requests are satisfied through multiple (if necessary) calls to BLSUPUTA, each call resulting in the generation of three blank lines. Finally, the message text is transmitted to the active destinations. If the CAPS keyword was entered, the text is converted to uppercase prior to transmission.

Dump Directory Handling Subcommands

The dump directory subcommands manage the dump data cross-reference records, symbol table records, and storage map records.

DROPDUMP Subcommand

The DROPDUMP subcommand drops all records associated with a particular dump from the dump directory. The processing of the subcommand is done by the BLSRDUDE and BLSRDUDR modules.

BLSRDUDE starts the processing by parsing the subcommand. If no data set name was specified in the subcommand, the default data set is used. If there is no default data set name, BLSRDUDE issues a diagnostic message.

If the dump data set is open (which will occur only if the default data set is being processed), a copy of the dump's RD (data set) record is made from the dump's BLSUZZ6 control block, and the dump is closed by BLSRDUCC (which erases the BLSUZZ6 control block). Otherwise the dump is already closed and it is necessary to read the dump's RD record directly from the dump directory. If no such record exists, a message is issued (BLS18135I), and the subcommand terminates.

The RD record is marked to show termination is in progress and rewritten in the dump directory.

BLSRDUDR then erases from the dump directory the PH (program history), ES (equate symbol), SA (storage address), RA (address space), and RD (data set) records for the dump (in that order). Erasure of the RD record is saved for last so that if processing is abnormally terminated it is possible to restart DROPDUMP processing later.

LISTDUMP Subcommand

BLSRDULS is the module that processes the LISTDUMP subcommand.

BLSRDULS reads the dump name (RD) records from the dump directory and displays the dump data set names.

After parsing entered operands and setting routing and flag indicators, a GENCB is executed to create an RPL, using ZZ1ACBP in the common area to point to the ACB. BLSUVSCR is called to do the GENCB. BLSUVSPO is called to point to the first RD record. Then each RD record in the dump directory is read via a call to BLSUVSGE, and the dump data set information is displayed.

LISTSYM and DROPSYM Subcommands

The LISTSYM and DROPSYM subcommands list and drop ES (equate symbol) records from the dump directory. (See the description of ES records under “Symbol Management” later in this section.) The processing of the LISTSYM and DROPSYM subcommands (modules BLSRLSYM and BLSRESDR, respectively) is essentially identical.

Both modules parse the subcommand (IKJPARS), constructing a chain of PDEs (parameter descriptor elements), each element of which corresponds to a symbol or symbol range specified in the subcommand.

The equate symbol records for the current dump are read sequentially from the cluster. Only one pass is made through the records. To improve performance the beginning and end of the symbol table are ignored if possible.

Each record in the symbol table is examined to see if it is “interesting.” In the case of LISTSYM all ES records (symbols) are “interesting.” For DROPSYM though, only “droppable” symbols are “interesting,” unless the PURGE keyword is specified in the subcommand. If PURGE is specified, the symbol is dropped in all cases.

All records in the symbol table are compared against all elements in the chain of symbol PDEs. Any PDE that matches the record is marked “found” to show that a symbol has been found within its bounds. In addition, if the record is an “interesting” record, the PDE is marked “used” and the record is dropped (for DROPSYM) or listed (for LISTSYM).

After all the symbol records have been processed, the “used” and “found” markings on the PDEs are used to produce diagnostic messages.

The modules used to implement these functions are as follows:

- BLSRLSYM/BLSRESDR - These modules parse the subcommand and generate the RPL necessary to sequentially read a dump directory. They also contain two internal procedures that identify “interesting” records (TESTSYM) and list or drop them (DOSYM).
- BLSRSYRB - This module determines how much of the beginning and end of a symbol table can be ignored given the chain of symbol PDEs as input. It also initializes all symbol PDEs (ranges) as not “used” and not “found.”

- BLSRESRU - This module is used by both LISTSYM and DROPSYM to produce diagnostic messages about unfound/undropped symbols given the chain of symbol PDEs.

EQUATE Subcommand

BLSREQU is the main module that processes the EQUATE subcommand.

BLSREQU associates a symbol with an address expression, and writes an equate symbol record with the information to the dump directory.

After parsing entered operands, BLSRADDR is called to resolve the address expression and set the fields of an equate symbol record in storage.

Next, the symbol name is filled in, the DROP/NODROP keyword option is analyzed and the proper setting made to the equate symbol (ES) record image, and BLSRESAR is called to write the record.

RENUM Subcommand

BLSRESRE is the main module that processes the RENUM subcommand.

BLSRESRE rennumbers the stack symbols for a dump.

BLSRESRE performs the following processing:

- After parsing entered operands, calls BLSRDUSO to prepare the dump for use.
- Calls BLSRESRN to renumber the stack for the dump.

STACK Subcommand

BLSRESST is the main module that processes the STACK subcommand.

BLSRESST associates a symbol with an address expression and writes an equate symbol record with the information to the dump directory.

BLSRESST performs the following processing:

- After parsing entered operands, calls BLSRADDR to resolve the address expression and to set the fields of an equate symbol record in storage.
- Fills in the symbol name.
- Analyzes the DROP/NODROP keyword option.
- Makes the proper setting to the equate symbol (ES) record image.
- Calls BLSRESSS to write the record.

LISTMAP, DROPMAP, and SCAN Subcommands

The LISTMAP, DROPMAP, and SCAN subcommands list, drop, and validate dump directory SA (storage address) records. The modules that process the subcommands are BLSRSALI (LISTMAP), BLSRSADE (DROPMAP), and BLSRSCAN (SCAN). The processing of the subcommands is similar.

All the subcommands parse the subcommand line (module IKJPARS). If any of the TEST, ROUTING, or FLAG keywords are specified, the corresponding field of the BLSUZZ2 control block is set.

The range of addresses to be processed is resolved in the following way:

- If the RANGE keyword has been specified the processing is restricted to the SA records of one address space. The range of addresses as well as the address space are determined by BLSRADDR.
- If no RANGE keyword is specified, but one or more address space keywords are (such as ASID, REAL, HEADER), then the processing is restricted to all addresses of one address space. The address space is determined by the BLSRADD2.
- Finally, if both the RANGE and address space keywords are omitted, all addresses in all address spaces are examined.

The modules read through the dump directory according to the range (SCAN may make multiple passes), and process each record according to which subcommand was issued. Before reading each record the modules check for an attention interrupt, and exit with a return code of 12 if one occurs.

In the case of SCAN, each eligible SA record is used to initiate a scan probe. A record is eligible if validity checking is incomplete for the record, and if a scan routine exists to do the checking. In addition, the record cannot have been generated during the current pass through the range by another scan probe. BLSRSAG does the scan probe. Diagnostic messages are issued if no records are found in the range, or if the scan probes produce no new results.

LISTMAP rescans and displays SA records. The RESCAN keyword forces a record to be rescanned (invoking the scan routine (BLSVxxxx series) directly) provided validity checking for the SA record has been completed (fields and pointers checked), a scan routine exists, and the return code for the validity check is not less than the current FLAG setting. The VERIFY keyword displays SA records and storage in accordance with the DISPLAY options currently in effect (BLST01). Only records for which validity checking has been started are displayed. A count of total records examined, rescanned, verified, and totally unchecked is kept and displayed in the summary message.

DROPMAP deletes all records within the specified range, and displays a count of total records dropped.

Parse Validity Checking

IPCS prefers to use IKJPARS to perform all syntactic checking of operands. However, frequently this is not feasible. IKJPARS will, for example, locate an INTEGER operand but will not restrict the range of values that the operand may describe. If IPCS does restrict the range of values, IKJPARS is asked to pass control to an IPCS validity checking subroutine whenever an operand is located. The IPCS subroutine will verify that the operand is valid or will inform IKJPARS via return codes 4 or 8 that it should be considered invalid so that IKJPARS may prompt for one that is.

ISPF Dialog Programs

Logical Screen Task Initialization

A logical screen task variable (see “BLSUZZ2” on page 406) is acquired and added to the IPCS chain of task variables by dialog program BLSG. The task variable provides for continuity between IPCS requests processed under the same logical task.

Processing IPCS Procedures

IPCS subcommands and CLISTs are processed by dialog program BLSGSCMD. An ECT (macro IKJECT) is generated by stacking a terminal element. The text passed to BLSGSCMD is passed to BLSUSCM1 which uses IKJSTCK to make it available to IKJPTGT later. Attention (STAX) exit BLS9MOI1 is established to permit monitoring and cancellation of the command or CLIST. Then BLSUMON is called to process the request. A zero-length “mode message” is passed to BLSUMON.

Dump Viewing Dialog

The dump viewing dialog provides full-screen dump viewing, using ISPF facilities accessed via service routine ISPLINK.

The full-screen dump viewing dialog consists of ISPF panels (BLSPaaaa in SYS1.SBLSPNL0), messages (BLSMann in SYS1.SBLSMSG0), variables, and six dialog modules, BLSLDISP, BLSLDSTG, BLSLIADS, BLSLIDSN, BLSLFISE, and BLSLPTRS. See Figure 27 on page 84 for the relationships among these elements.

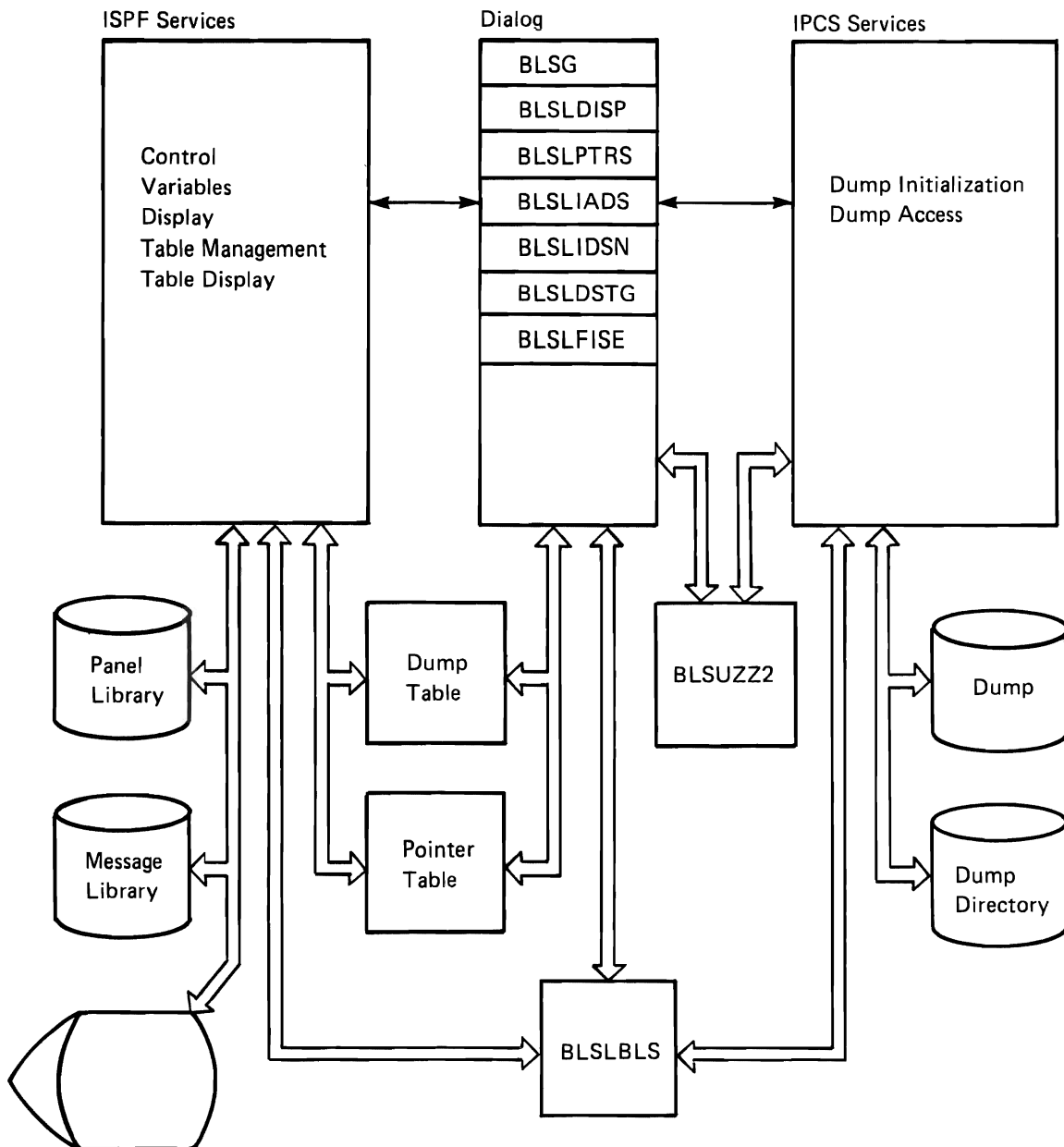


Figure 27. Dialog Overview: Relationship of Subsystems

A brief discussion of the six dialog modules follows.

BLSLDISP controls the operation of the dialog through the following processing:

- Verifies that the dialog task is prepared for IPCS by dialog program BLSG.
- Initializes the dialog work area, BLSLBLS, and identifies many of its fields to ISPF as function pool variables that allow the dialog and ISPF to exchange information throughout the dialog.
- Displays the dialog entry panel (BLSPOPT) to determine the source of data and, optionally, the address within which that source of the data is to be displayed. BLSLDISP calls BLSLIDSN to verify the input from the entry panel, BLSLPTRS, if a pointer list is to be displayed, and BLSLDSTG, if storage is to be displayed.

Processing of the entry, pointer, and storage panels is done repeatedly until the user requests the termination of the dialog.

- Deletes the definitions of ISPF function pool variables and returns to its caller.

BLSLIDSN processes input from the dialog entry panel (BLSPOPT) for BLSLDISP.

BLSLPTRS displays the pointer stack through the following processing:

- Establishes the ISPF function pool variable, BLSPPTR by
 - Determining the size of the dynamic area on the pointer panel (BLSPDISP) that will be associated with BLSPPTR via the ISPF PQUERY service.
 - Obtaining a block of storage large enough for BLSPPTR plus some control information for each line to be formatted in BLSPPTR.
 - Identifying the block of storage to ISPF as BLSPPTR.
- Formats BLSPPTR and other function pool variables to describe pointers associated with locations in storage.
- Displays the pointer panel, then processes requests to manipulate the stack, to display other stack entries, and to display the storage described by a pointer.

Note: BLSLIADS is used to process address space text; BLSLDSTG is used to display storage.

- Performs clean up. BLSLPTRS deletes BLSPPTR (the definition of the function pool variables), frees the related storage, and returns to its caller, BLSLDISP, when CANCEL, END, or RETURN primary commands are entered.

BLSLDSTG displays dump storage through the following processing:

- Establishes ISPF function pool variable, **BLSDATA** by
 - Obtaining the size of the dynamic area from the storage panel (**BLSPDISD**) that will be associated with **BLSDATA** via the ISPF **PQUERY** service.
 - Obtaining a block of storage large enough for **BLSDATA** plus some control information for each line to be formatted in **BLSDATA**.
 - Identifying the block of storage to ISPF as **BLSDATA**.
- Formats **BLSDATA** and other function pool variables to describe storage at the requested address.
- Displays the storage panel, updates the definition of symbols, **CURSOR** and **X**, and processes requests to manipulate the stack and to display data at another address.
- Performs cleanup. **BLSLDSTG** deletes **BLSDATA** (the definition of the function pool variable), frees the related storage, and returns to its caller, **BLSLDISP**, when **CANCEL**, **END**, or **RETURN** primary commands are entered.

BLSLIADS translates address space keywords to IPCS' internal format. When it is entered at its alternate entry point (**BLSLIADX**), **BLSLIADS** also processes a positional address expression preceding the address space keywords.

BLSLFISE processes **FIND** and **RFIND** primary commands for **BLSLDSTG**.

Common Service Functions

This section describes the method of operation for each of the the following common service functions:

- Data Access Services
- Print and Terminal Services
- Message Text Build and Routing
- Common Exit Services
- Dump Access
- Dump Initialization
- Storage Map Management
- Scan Routines
- Symbol Management
- Find Routines
- Address Space Mapping
- Address Resolution
- Dump Directory Handling
- Dump Data Formatting

Data Access Services

The data access services function handles all of the I/O for the problem and data management subcommands of IPCS. The purpose of the function is to isolate in one area of the IPCS structure the functions which permit access to the problem directory and data set directory, in order to ensure the data integrity of these shared resources.

Data access services provides, in addition to the I/O functions of allocation, opening, reading, writing, closing, and freeing, resource serialization constraints which must be adhered to by the problem and data management subcommands.

Data Access Services Overview

Data access services provides the following functions:

- Reserves and releases data base resources.
- Supports the access mode integrity.
- Simulates the read/write multi-access mode.
- Ensures data integrity other than data content errors.
- Provides the interface between other IPCS functions and the IPCS data base.

Figure 28 on page 88 presents a functional block diagram of data access services. Data access services is comprised of three major functions: data access, allocation, and deallocation. Data access services maintains interfaces to the following system access services:

- VSAM access method for handling VSAM data sets.
- QSAM access method for handling sequential non-VSAM data sets, including access to members of a partitioned data set.
- MVS/XA dynamic allocation function for connecting requestors to data set resources.

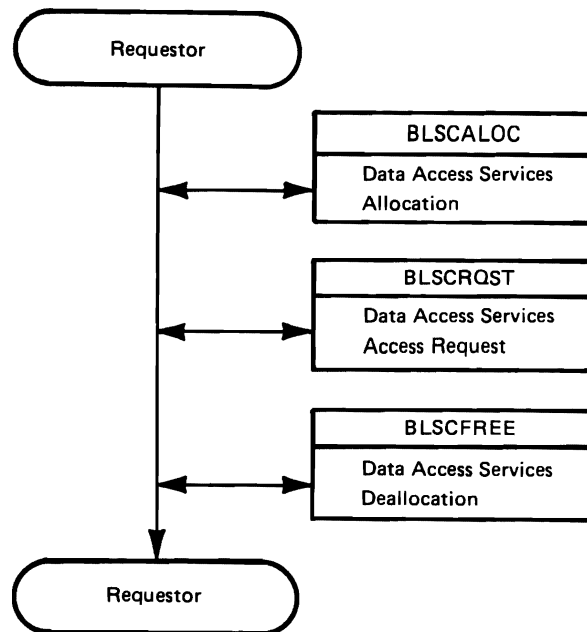


Figure 28. Data Access Services Overview

Invoking Data Access Services

The data access services function is invoked by IPCS subcommands. The first function invoked by these subcommands connects the requestor to the data set, creates a data access services control block (BLSDMCB) and returns its address to the requestor. The BLSDMCB address is required by all of the other data access services. A summary of the services follows:

- Obtain space for and initialize a DMCB, obtain DASD space for a new data set, reserve the data set for the requestor, associate a user supplied file name with the data set or supply a system generated file name for the data set, and connect the requestor with the DMCB for use by the requestor's program when accessing the data set.
- Create a DMCB and connect the data set and file name previously allocated by JCL or the TSO user to the DMCB for the requestor.
- Conditionally obtain space for and initialize a DMCB, obtain DASD space for a new data set, reserve the data set for the requestor, associate a user supplied file name with the data set or supply a system generated file name for the data

set, and connect the requestor with the DMCB for use by the requestor's program when accessing the data set.

- Free DASD space as requested, release the data set from the requestor, and FREEMAIN the DMCB.
- Connect a data set to the requestor's program via the DMCB.
- Disconnect a data set from the requestor's program.
- Change access-related parameters in the DMCB without issuing a data access request.
- Retrieve a record from the data set associated with the DMCB.
- Position a data set associated with the DMCB.
- Add or update a record in the data set associated with the DMCB.
- Delete a record previously obtained for update from the data set associated with the DMCB.
- Terminate a request, usually to release a record obtained for update.

Data Set Allocation

Before allocating a data set, data access services examines the parameters passed from the requester. The parameters include a list of allocation override parameters.

Figure 29 on page 91 describes the operation of the allocation function.

This function has two purposes:

- To create a new IPCS data set.
- To connect a requester to an existing data set.

The allocation function (BLSCALOC) first creates the DMCB, then it controls the execution of other allocation routines. The first routine (BLSCAINT) prepares data access services to do the IPCS I/O by initializing the new DMCB. The next routine (BSLCABLD) builds an allocation-override list using parameters passed from the requester. The override parameters are merged (BLSCAMER) with the existing default parameters to create the dynamic allocation parameter list in the DMCB. The default parameters are established in the allocation model (BLSCAMOD) supplied with IPCS.

The operation of connecting the requester to the data set depends on the type of data set, as follows:

- If the allocation request is for a new VSAM data set, the access method services interface (BLSCAAMS) is called to obtain the space. The dynamic allocation controller (BLSCADYN) uses the MVS/XA dynamic allocation function (SVC 99) to connect the requester to the data set.
- If the request is for a new non-VSAM data set, the dynamic allocation controller (BLSCADYN) obtains the space as well as connecting the requester.

- If the allocation request is for existing data, after initialization and construction of the allocation parameters, the dynamic allocation controller (BLSCADYN) connects the requester to the data set.

Control is returned to the requester after the data set connection is made.

The requester uses the DMCB to obtain the entry point for the data access function.

If any errors are encountered during allocation, error messages are constructed (BLSCANAL) for the requester. If there are severe errors, no allocation is done, no DMCB is built, and register 0 contains a pointer to the queue of error messages.

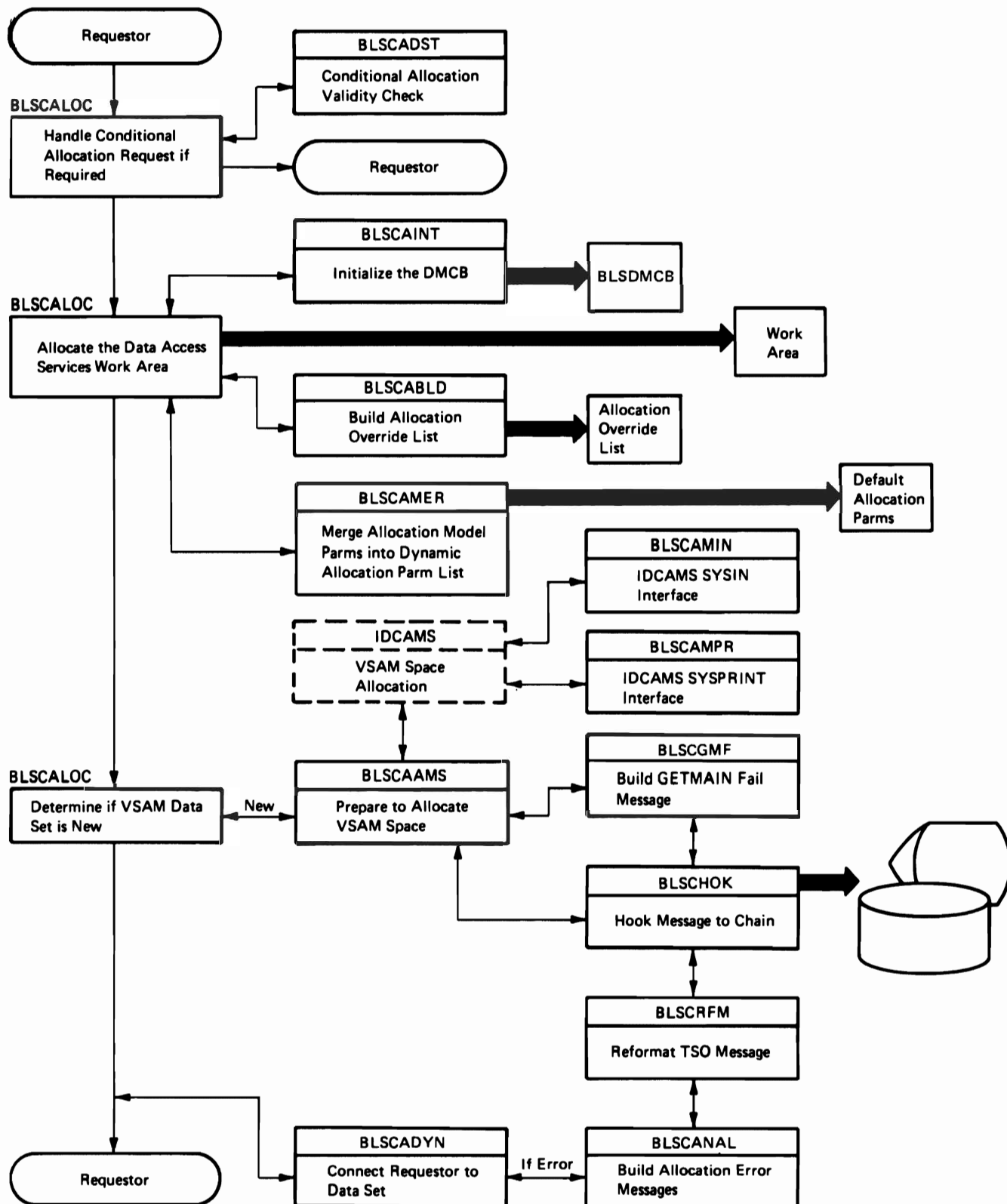


Figure 29. Data Access Services Allocation

Data Set Access Requests

After the data set has been allocated and connected, control returns to the requester. The requester receives the address of the DMCB. The requester then requests the service he desires, including the DMCB pointer as a parameter. For example, the requester may gain access to a connected data set, then retrieve a data record. Figure 30 on page 94 shows the activity during an access request.

Data Set Open: A data set resource is opened for the requester. Data access services ensures that the resource has been previously allocated. Based upon the access mode, the OPEN request indicates the INPUT option or the UPDATE/OUTPUT option. The requested mode of access is validated against the access mode declared in the allocation request. In addition, if the data set has an access mode of multiple read/write, data access services issues the ENQ macro, required by the high-contention protocol (see Appendix C for details of this protocol).

Data Set Get, Put, Point, Erase, and End: Data access services ensures that the request is allowed, that is, that the data set resource has been allocated and opened, and that the access mode permits the read or write request. The QSAM DCB or VSAM RPL is then initialized with the request parameters and the request is performed by the system.

Data Set Close: The data set resource is closed for the requester. If the data set has an access mode of multiple read/write, data access services issues the DEQ macro, required by the high-contention protocol (see Appendix C for details of this protocol).

Data Access Operation: The first data access module to receive control is BLSCRQST, the request module. It receives as input the DMCB. Data access uses the DMCB to establish a path between the allocated data set, the input/output data buffer, and the requester. The request module initializes the data access function by validating the request code, determining the type of request, and invoking the appropriate executor module to perform the request.

The I/O access connect routines (BLSCOPEN, BLSCCLSE) are used to open or close the data set. The open and close modules:

- Establish the ESTAE exit to the recovery module (BLSCRECV).
- Determine if the request is for VSAM or QSAM.
- For high-contention data sets, they validity check the request (BLSCVALT) and issue the data set enqueue (BLSDENQ0) or dequeue (BLSDDEQ0).
- Issue the OPEN or CLOSE request to the system.
- Cancel the ESTAE exit.
- Return to the request module (BLSCRQST).

When the requester issues a BLSGET request, the get module (BLSCGETT) receives control. The get module determines whether the type of data requested is either VSAM or non-VSAM. If VSAM, the data could be either entry-sequenced or key-sequenced. The get module then obtains the data record and moves it into the data buffer pointed to by the DMCB.

A similar operation is followed for a put request (BLSCPUTT) to store a record in a data set, an erase request (BLSCRESE) to erase a record from a data set, and an end request (BLSCENDDD) to cancel an operation that has been started.

To find a specific key-sequenced item of data, the requester can issue a BLSPNT request to obtain a pointer to the location of the data. When the pointer is returned, the requester can issue a BLSGET request to read the data.

The five request executor modules (BLSCGETT, BLSCPUTT, BLSCPOIN, BLSCRESE, and BLSCENDDD) all use the following criteria to determine which form of request to issue:

- Access method - either QSAM or VSAM.
- File type - ESDS (sequential) or KSDS (keyed).
- Access type - Sequential, direct, or relative record request modifier.
- Request validity - Is the request supported by access method and file type sequence errors?

If data access operations are being performed on a VSAM data set, BLSCSETT, the control block modifier module, is invoked. This module sets up the request parameter list (RPL) for VSAM. The request executor module issues the request to VSAM, and analyzes the return code before returning to BLSCRQST.

The control block modifier (BLSCSETT) is invoked by the executor modules for VSAM file requests. It may also be invoked by a data access services requester via BLSCRQST as a BLSSET request. The control block modifier:

- Examines the DMCB for any changes to parameters which require a change to the RPL.
- Builds a parameter list for the VSAM MODCB macro.
- Issues the MODCB macro to change the RPL.
- Returns to the calling executor module.

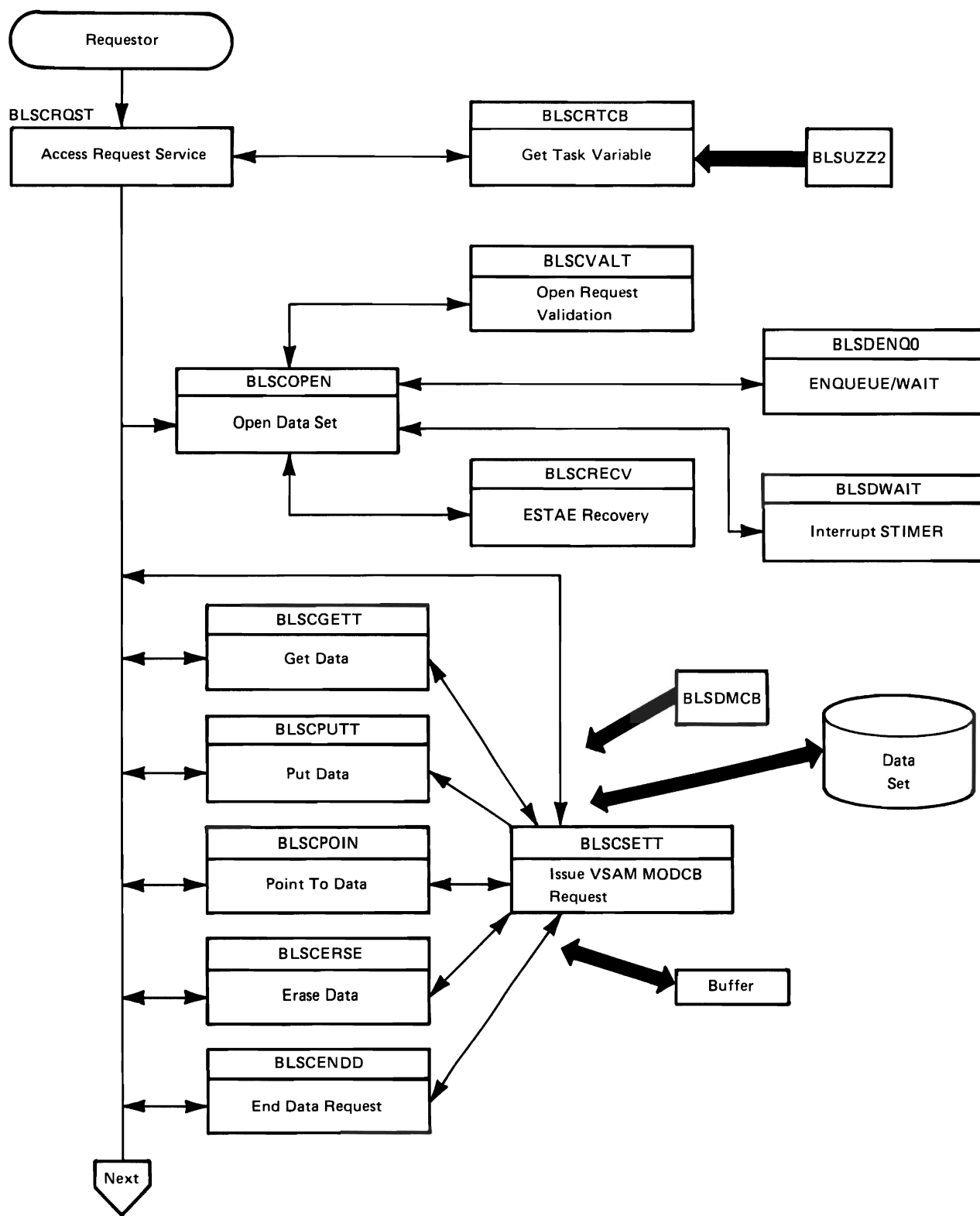


Figure 30 (Part 1 of 2). Data Access Services Access Request

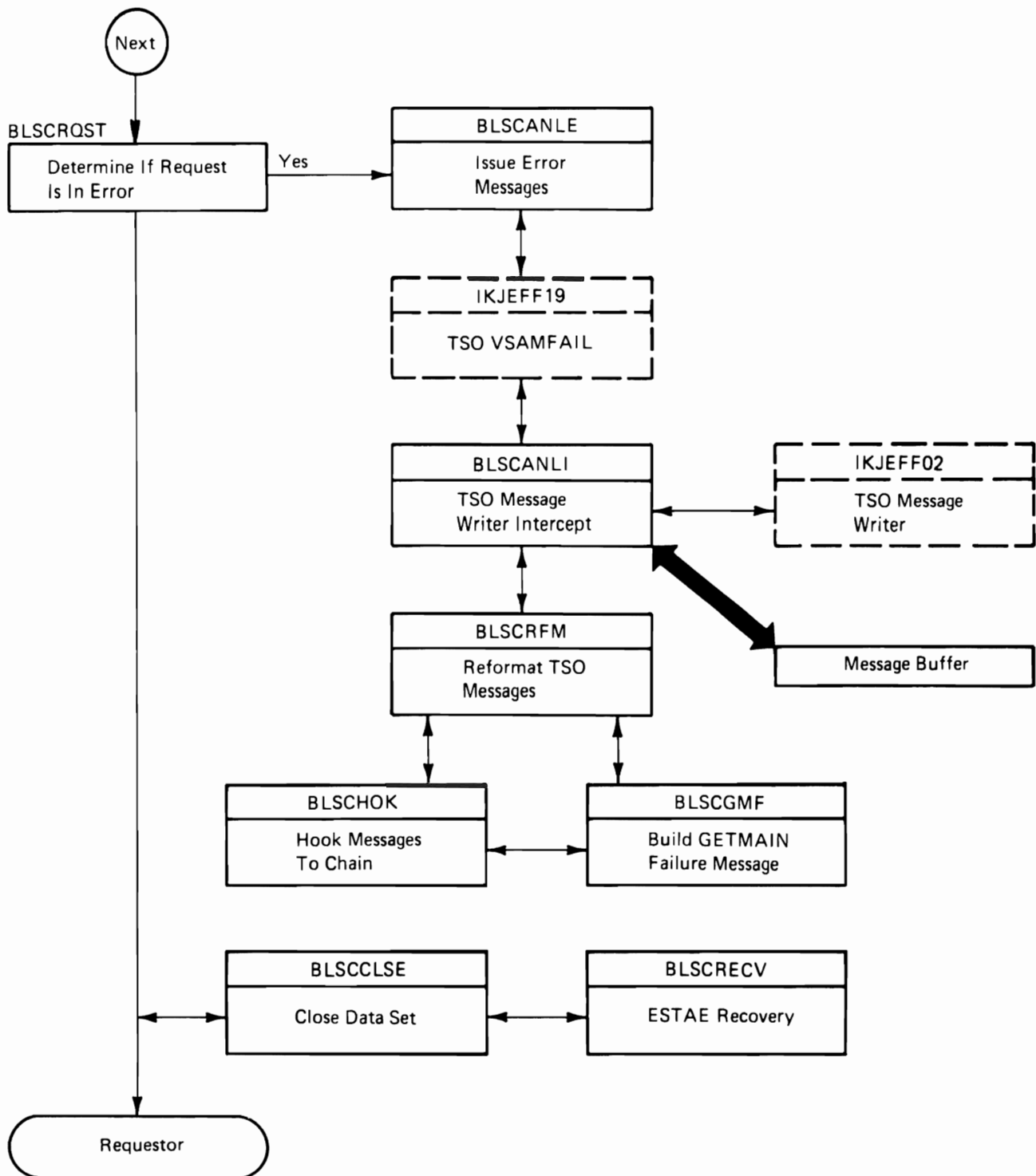


Figure 30 (Part 2 of 2). Data Access Services Access Request

Data Set Deallocation

The data set resource is freed for the requester. In addition to the deallocation, data access services frees all resources that were obtained for the allocation of the data set.

Figure 31 on page 97 shows the operation of the deallocation function. BLSCFREE is invoked to perform the deallocation function. The data set name and its disposition parameters are passed in the DMCB. However, if the allocation disposition is overridden, then the disposition is passed in the FREE keyword parameter list. The requester is disconnected from the data set (BLSCFDYN). If the request is for a VSAM data set, and the disposition is DELETE, control passes to BLSCFAMS which provides an interface to IDCAMS to delete the data set.

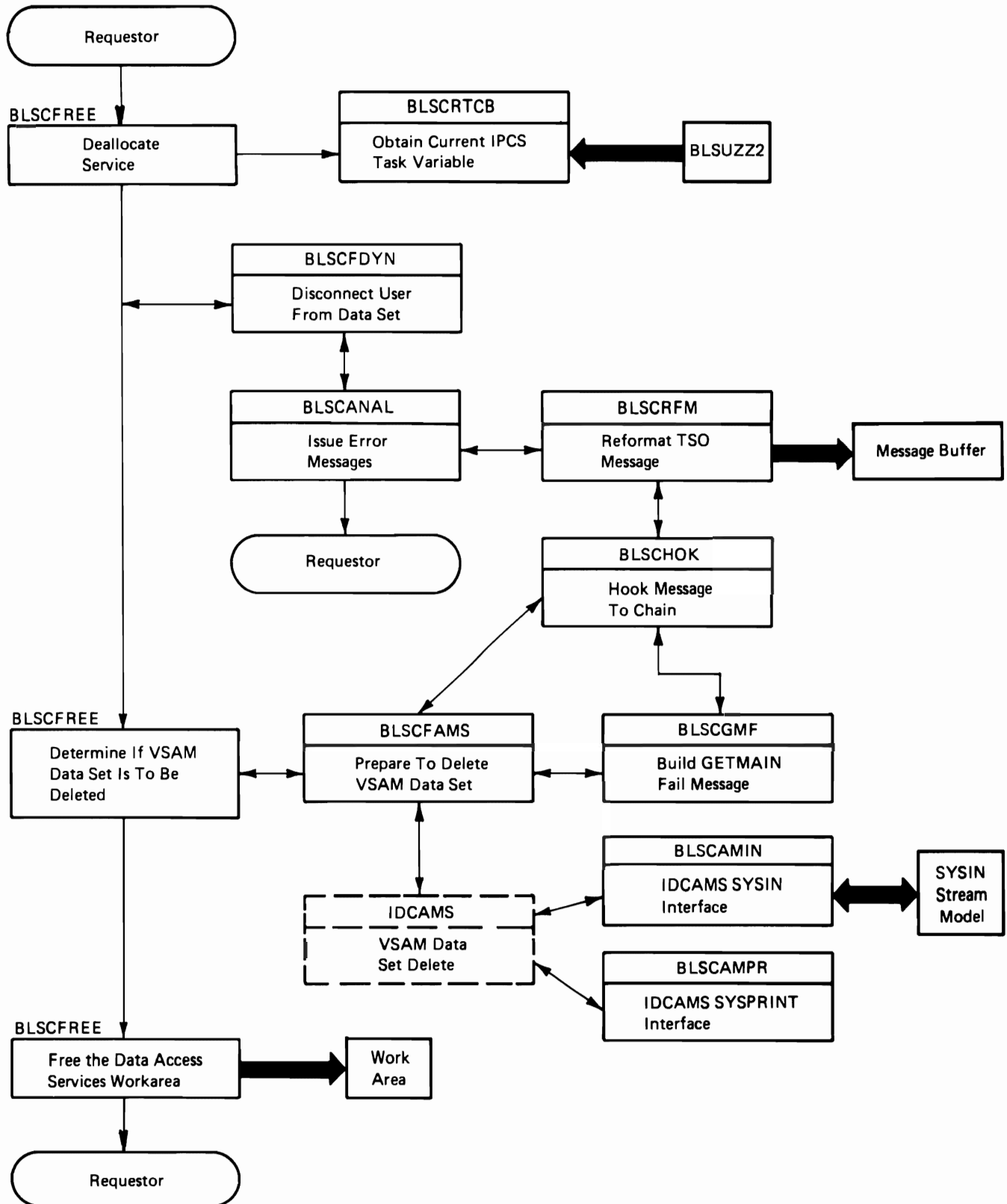


Figure 31. Data Access Services Deallocation

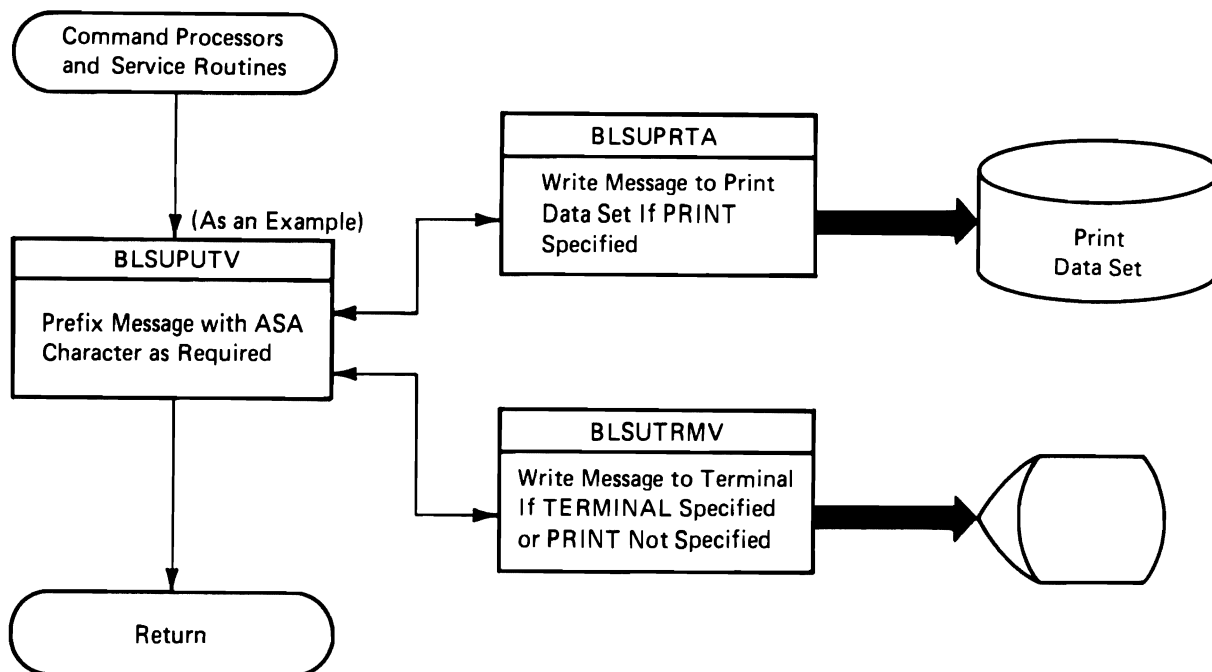


Figure 32. Print and Terminal Services

Print and Terminal Services

The print and terminal service routines perform IPCS message processing functions. These routines are divided into two areas: routing functions (routers) and transmitting and control functions. The message to be processed is supplied to these routines as a calling sequence parameter.

Message Handling

The design of the IPCS print and terminal service routines is intended to reduce and localize the extent to which IPCS coding is dependent upon:

1. Message destination.
2. Access method interfaces.

The destination of messages is determined by the values of two subcommand keywords, `TERMINAL/NOTERMINAL` and `PRINT/NOPRINT`, from which two fields in the BLSUZZ2 control block are set during subcommand processing. These fields indicate whether messages are to be directed to the printer, to the user's terminal, or to both.

Messages intended for the user's terminal are transmitted via standard interfaces to the TSO I/O service routines for terminal I/O, while standard QSAM interfaces are established when the destination is the printer.

Substantial freedom from these two dependencies is achieved within IPCS message issuing routines through the availability of a set of message routing routines (BLSUPUTx), and a set of message transmitting and control routines (BLSUPRTx, BLSUTRMx). Message issuing routines construct messages without regard to message destination, and with limited regard to access method interface requirements. Then, having considered the following four message characteristics, the message issuing routine invokes the appropriate message routing or message transmitting routine:

1. Is destination independence desired?
2. Is an ASA control character present in the message?
3. Is a message ID present in the message?
4. Are multilevel or multiline messages required?

Routing Routines (Routers)

Each router accepts messages which conform to a specific combination of the last three characteristics shown above. For example, router BLSUPUTV processes variable-length messages which, optionally, are preceded by a single-space ASA control character. A router does not exist for each possible combination of message characteristics, but routines corresponding to the subset of such combinations as are used by IPCS message issuing routines are provided. It is the responsibility of the calling function to invoke the correct router.

The purposes of a router are to perform any necessary message structuring and to invoke the appropriate message transmitting routine as indicated by the currently active message destinations. The processing performed by router BLSUPUTV is shown in Figure 32 on page 98 as an example of router processing. Other routers operate in a similar manner.

Transmitting Routines

The transmitting routines are invoked by the routers and contain the access method interfaces. They also are called directly by IPCS message issuing routines whose messages are in the proper format and to whom the active message destination is known.

There are two groups of transmitting routines: one group for printer devices, the other for terminals. Within each group, each transmitting routine processes messages of a specific type and format. Thus, BLSUTRMA transmits messages that exist in RECFM=VA format to the terminal, while BLSUTRMN transmits multi-line RECFM=VA messages to the print file.

Control functions such as print page eject and page titling (BLSUPRTA), and the insertion of blank lines on the terminal, also are performed by the transmitting routines. Additionally, a separate routine, BLSUPRTT, allows a message issuing routine to establish a printer page subtitle.

Message Text Build and Routing

BLSDMSG0 performs the function of building and routing all problem management, data management, and data access services messages. The message skeletons are stored in a message CSECT (BLSDMSGs); the caller indicates the

message to be produced by its message number. If the message number supplied by the caller is a minus one, message construction is bypassed.

Message construction consists of merging any message inserts supplied by the caller into the message skeleton. A flag parameter indicates whether trailing blanks are to be removed from each insert prior to its being combined with the message skeleton. Following the construction of the current message, it is inserted either at the beginning (LIFO) or at the end (FIFO) of the message queue supplied by the caller. LIFO or FIFO queueing is indicated by a flag parameter.

The routing flag parameters are checked to determine whether the message queue, now also containing the new message if one was constructed, is to be routed. If routing was requested, either BLSUPUTN (both printer and terminal), BLSUPRTN (printer only), or BLSUTRMN (terminal only) is called to output the message. This routing only refers to the scope of the routing available; actual routing is controlled by the SETDEF default value settings currently in effect. If no routing is requested, none of these modules are called.

If routing was performed, or if the input message number was minus one, the messages on the message queue are FREEMAINED. If no routing was performed and the input message number was not minus one, the messages on the message queue are left intact and the queue is returned to the caller.

Common Exit Services

The following list contains the common exit services and the dump processing exit host that supports the service:

Exit services router	IPCS, PRDMP, SNAP
Storage access	IPCS, PRDMP, SNAP
Print	IPCS, PRDMP, SNAP
Format model processor	IPCS, PRDMP, SNAP
Control block formatter	IPCS, PRDMP, SNAP
Index	PRDMP
ECT	IPCS, PRDMP, SNAP
Get symbol	IPCS
Equate symbol	IPCS
Select ASID	IPCS, PRDMP, SNAP

For PRDMP's common exit services see the *Service Aids* publication and for SNAP's common exit services see the *User Exits* publication. IPCS' storage access, print, and format common exit services are mentioned under PRDMP Exit Support Subcommands section. A discussion of the other services follows.

Exit Services Router

The exit services router, BLSQROUT, provides a routing mechanism for exits invoked from IPCS, PRDMP, and SNAP. An exit can invoke a particular service by calling BLSQROUT, whose address is in the ADPLSERV field in BLSABDPL, and by passing BLSQROUT the address of a parameter list that contains:

- The address of the ABDPL.
- The address of a fullword containing a service code. The BLSABDPL mapping includes constants defining the service codes, which indicate the service requested. (See Figure 33 on page 101).

- The address of the parameter list for the requested service. The BLSABDPL mapping includes mappings of each service parameter list. For the print and index services, all parameters are in the BLSABDPL and, therefore, there is no additional parameter list.

CONSTANT	SERVICE
ADPLSACC	Storage Access
ADPLSPRT	Print
ADPLSFMT	Format Model Processor
ADPLSCBF	Control Block Formatter
ADPLSNDX	Index
ADPLSECT	Ect Exit
ADPLSGTS	Get Symbol
ADPLSEQU	Equate Symbol
ADPLSSEL	Select ASID

Figure 33. Service Code Constants and Their Related Services

Note: The print and storage access exit services can be called as discussed through the exit services router or they can be called directly via fields in BLSABDPL. The format service can only be invoked via field ADPLFRMT in BLSABDPL.

BLSQROUT uses a work area called the services router area (BLSQSRA). BLSQSRA contains a table of service routine addresses set up by the dump exit host (IPCS, PRDMP, or SNAP). BLSQROUT uses the service code to find the service routine address. If the entry is zero, the service is not supported by the host and BLSQROUT returns with a return code of 0 in register 15 and 4 in the SRACODE field to indicate that no processing was performed. If the entry is nonzero, BLSQROUT calls the service, and on return, passes back its return code in register 15. The BLSQSRA also contains information needed by the exits for input (for example, the address of the ECT for the ECT exit service) and for communication. The ADPLSRA field within the BLSABDPL contains the address of BLSQSRA.

Module, BLSQEXTI, sets up the exit interface for IPCS, PRDMP and SNAP. BLSQEXTI loads the exit services that the host indicates are supported and fills in the BLSABDPL and BLSQSRA with the addresses of the supported exit services. If the service cannot be loaded, BLSQEXTI issues a message and sets the return code to 12.

Control Block Formatting Service

This service allows an exit program to use an acronym to request the formatting of a control block. No knowledge of the control block's structure is needed. Only the location of the control block is required. The service uses a table of acronyms called the control block acronym table (CBAT) to associate acronyms with format

models and format modules. The following control blocks are supported by this service:

ASCB	LCCA	SRB
ASXB	LLE	STKE
CDE	PCCA	SUPVT
CSD	PSA	SVT
CVT	RB	TCB
EED	RTCT	TIOT
FRRS	RTM2WA	XSB
IHSA	SCB	XTLST

Simple control blocks are formatted via the format model processor using a format model.

BLSQCFMT is the module that provides the control block formatting service. BLSQCFMT is invoked by exit programs via BLSQROUT. BLSQCFMT requires as input the ABDPL, an initialized ADPLPFMT parameter list, a specified control block acronym, either the location of the control block in the dump, or the location and length of the control block in a storage buffer, and the view control to be used. The ADPLPVCL field contains the view control options, which permit the caller to select specified fields to be formatted. See the ADPLPVCL field in “BLSABDPL” on page 326 for the various bit settings that reflect the view control.

When formatting control blocks, various options are available, such as suppression of offsets, suppression of a header, a blank line count, starting offset, and line mode. See the *Macros and Facilities* publication for additional information about the BLSQMDEF and the BLSQMFLD macros, which permit formatting of control blocks to the user’s own specifications.

Results of processing are passed back to the caller in the ADPLPFMT.

The following steps are performed:

CBAT lookup: The acronym specified by the caller is compared with entries in the control block acronym table (CBAT) to find the names of a model and/or the name of a program module.

Loading: The model and/or the program module is loaded.

Pass control: If a program module was specified, the module is given control to continue the formatting process. If a model was specified instead of an acronym, control is given to the format model processor BLSQIFMT via BLSQROUT, passing it the ADPLPFMT parameter list.

Format Model Processor Service

BLSQIFMT is the module that processes the format model and prints the results. The model contains the control block fields that are formatted.

The control block formatting service or any exit program can invoke BLSQIFMT via BLSQROUT. BLSQIFMT requires as input the ABDPL and the ADPLPFMT parameter list.

The following functions are performed:

Conditional loads: If the pointer to the model is zero, indicating that the model is not in storage, the model is loaded, using the model name specified by the caller. If the pointer to the data in storage is zero, indicating that it is not in storage, the data is read from the dump, using the dump address specified by the caller and the control block length from the model. The caller must have established ADPLASID prior to calling.

Options processing: If the caller specified acronym checking, the acronym information in the model is used to compare the acronym in the control block with the acronym in the model. If the comparison fails, a return information bit is set, and if error messages are not suppressed, a message is printed.

Format the control block: Passes control to BLSQFORM to format the control block.

BLSQFORM is the module that produces the formatted control block from the image in storage and the control block model.

BLSQFORM will perform the following functions:

Initialize local variables: Control information is taken from the model, ABDPL, and ADPLPFMT to set local variables that control such factors as:

- Starting column
- Maximum line length
- Offset suppression

Formatting: The format model consists of a header containing information pertinent to the whole control block and an array of entries containing information pertinent to individual fields. The field oriented information is used to access the data, convert it to printable hexadecimal (if necessary), and merge it in the print buffer with field labels from the model. As each line is filled, it is printed. The model might contain subheaders that can appear in the output either in the next data location or on a new line. Flags in the field entries can specify calling an exit provided by the caller, starting a new line, or suppression of the label. The view control flags in the model control the selection or rejection of each field. The view control flags in the model are matched with the view control specified by the caller. Arrays within the control block of fixed dimension are processed automatically. Arrays of variable dimension are processed if the caller specifies a non-zero array count.

In addition to formatting control blocks, the format models and the format model processor can be implemented to display a variable mix of subheaders, to decode a flag byte, or to display extracted control block information.

Select ASID Service

This common exit service generates a list of pointers to ASCBs in a dump. The ASCBs must meet the selection criteria of at least one of the following keywords:

ASIDLIST
ALL
CURRENT
ERROR
JOBLIST
TCBERROR

The select ASID service permits selective address spaces to be processed.

BLSQSEL is the module that provides the select ASID service. The dump processing exit programs (IPCS, PRDMP, SNAP) call BLSQSEL via BLSQROUT.

The input parameter list specifies:

- The selection criteria
- Options
- A list of jobnames for JOBLIST
- A list of ASIDs for ASIDLIST

All dump accesses are processed using the dump access service.

The following steps are performed:

Setup processing: If the CVT pointer in BLSABDPL is not zero, the CVT is obtained from the dump. The ASVTMAXU field is obtained from which the actual length is calculated. A save area equal to the actual length of the ASVT is obtained via a GETMAIN macro and the ASVT is obtained and moved into that save area. Another work area is obtained for constructing output list entries for selected ASIDS.

Process the selection keywords: If the ALL keyword is specified, all entries in the work area corresponding to assigned ASVT entries are marked 'selected by all'.

If the CURRENT keyword is specified, and the dump is a standalone dump, the current address spaces are located as follows:

1. The CVTCSD is obtained and saved.
2. The CSDCPUAL field is obtained.
3. The status record for each active CPU is examined.
4. The primary and secondary ASIDs are saved from the status record.
5. The PSA is obtained using the prefix or 0. Real storage and information is saved:
 - a. PSAHLHI - lock indicator
 - b. PSAANEW
 - c. PSALOCAL - lock word
 - d. PSAAOLD
 - e. And the ASID is obtained from the ASCB at PSAAOLD
6. Primary and secondary ASIDs assignments are checked in the ASVT.
7. If the local lock is held, the CML ASID is obtained.

If the dump is an SVC Dump or SYSM Dump, the current ASID is obtained from the header record.

For both types of dump, the corresponding entries in the work area are marked 'selected by current', and the CPUID is inserted. Also, a symbol for the ASCB is added to the symbol table.

If the ASIDLIST keyword was specified, the entries in the list of ASID parameters are processed. Each entry can be either a single ASID or a pair of ASIDs specifying a range of ASIDs. The ASID is used to index into the work area to mark the entry 'selected by ASID'.

If the ERROR keyword, the TCBERROR keyword, or the JOBLIST keyword, was specified, then each assigned ASCB pointed to by the ASVT is examined.

For each ASCB accessed, these steps are done:

1. The equate symbol service is called to add a symbol to the symbol table for the ASCB.
2. If the JOBLIST keyword was specified, the jobname pointed to by ASCBJBNI or ASCBJBNS is compared with the list of jobnames in the input parameter list.

If a match is found, the corresponding entry in the work area is marked 'selected by joblist'.

3. If either the ERROR keyword or the TCBERROR keyword was specified, the ASCBMCC field is examined. If it is not zero, the corresponding entry in the work area is marked 'selected by error'.

If the JOBLIST keyword or the ASIDLIST keyword was specified, only the ASIDs selected by JOBLIST or ASIDLIST are examined for error conditions.

If ASCBMCC is zero and the TCBERROR keyword was specified, the TCBS are obtained one at a time and examined.

If any TCB contains a non-zero TCBCMP or TCBSNDY field, then the corresponding entry in the work area is marked 'selected by tcberror'.

The jobname associated with each selected ASCB is accessed and put into the output list entry. If ASCBJBNI and ASCBJBNS are both zero, or if storage access fails attempting to access the jobname, then '*UNKNOWN' is put in the jobname field for that ASCB.

4. When the dump access service returns a nonzero return code for an ASXB, indicating that the requested data is not in the dump, the corresponding entry in the work area is marked 'data not in dump'.

Construct the ASIDLIST and cleanup: The selected ASIDs are counted to calculate the length of the ASIDLIST that is to be constructed, and storage is obtained to build the ASIDLIST.

A header is constructed that contains:

- The length of the ASIDLIST
- The storage subpool of the ASIDLIST
- The number of entries in the ASIDLIST
- A summary of conditions:
 - Some jobname(s) were not found.
 - Some ASID(s) in ASIDLIST were unassigned.
 - Some ASID(s) selected were not in the dump.

The work area entries are scanned, and for each selected ASID, an entry in the ASIDLIST is constructed, which contains the following information:

- Flag fields indicating:
 - The selection criteria met
 - Home, primary, secondary
 - Holds CML lock
 - ASCB accessed or not
 - ASCB access successful or not
- The ASID
- A pointer to the ASCB
- The jobname index if selected by JOBLIST
- The CPUID if selected by CURRENT
- The associated jobname
- A reserved area

Lastly, the address of the output list is placed in the input parameter list, the work area and the save area are released, and BLSQSEL returns to BLSQROUT, which in turn returns to the calling exit program.

ECT Exit Services

This service provides a routing mechanism for:

- One exit to invoke another exit or a group of exits
- Mainline IPCS or PRDMP processing to invoke an exit or a group of exits.

BLSQECT is the only module that processes the ECT service. BLSQECT performs the following processing:

- Checks to determine whether a single exit or a group of exits is to be called.

For a single exit

- If PRDMP is the dump processing host, BLSQECT searches the AMDPRECT for the specified exit name, such as DAEDATA.
- If IPCS is the dump processing host, BLSQECT searches the AMDPRECT for the specified exit name or the specified exit module name, such as DAEDATA, which represents the exit name and ADYHDFMT, which represents the exit module name.

For a group of exits

- When called by PRDMP or IPCS, BLSQECT searches the AMDPRECT for all the exits of a specified type. For example, BLSQECT checks the ADPLPETB bit in the ADPLPECT to determine which exits can perform additional processing after the TCBs are formatted.

Get Symbol Service

This common exit service's sole function is to obtain a properly initialized equate symbol data area. The dump processing exit host operating in an IPCS environment can invoke the get symbol service.

BLSQESGU is the module that processes the get symbol service. After receiving control in the addressing mode of the caller, with the caller's save area and symbol table record parameter addressable in that mode, BLSQESGU performs the following processing:

- Obtains a save area and a buffer for a symbol table record below 2²⁴
- Chains the caller's save area to itself
- Copies the caller's symbol table record into its buffer
- Calls BLSRESGU to process the request
- Copies the symbol record retrieved to the caller's buffer
- Issues the same return code that was received from the service routines that it called

Equate Symbol Service

This common exit service creates a symbol entry for major control blocks or data areas and places the entry with any associated attributes (such as address or length) into a symbol table. Dump processing exit programs operating in an IPCS environment can invoke the equate symbol service using module BLSQRROUT. The BLSRESSY macro must be initialized to the symbol information that is passed as a parameter. The equate symbol service reduces the access time for subsequent references to the same control block or data area.

BLSQESAR is the module that processes the EQUATE symbol service. After receiving control in the addressing mode of the caller, with the caller's save area and symbol table record parameter addressable in that mode, BLSQESAR performs the following processing:

- Obtains a save area and a buffer for a symbol table record below 2²⁴
- Chains the caller's save area to itself
- Copies the caller's symbol table record into its buffer
- Calls BLSRESAR to process the request
- Copies the symbol record retrieved to the caller's buffer
- Issues the same return code that was received from the service routines that it called

Dump Access

The dump access function obtains information from dumps on behalf of IPCS subcommands. Figure 34 shows the logic flow used when accessing dumps.

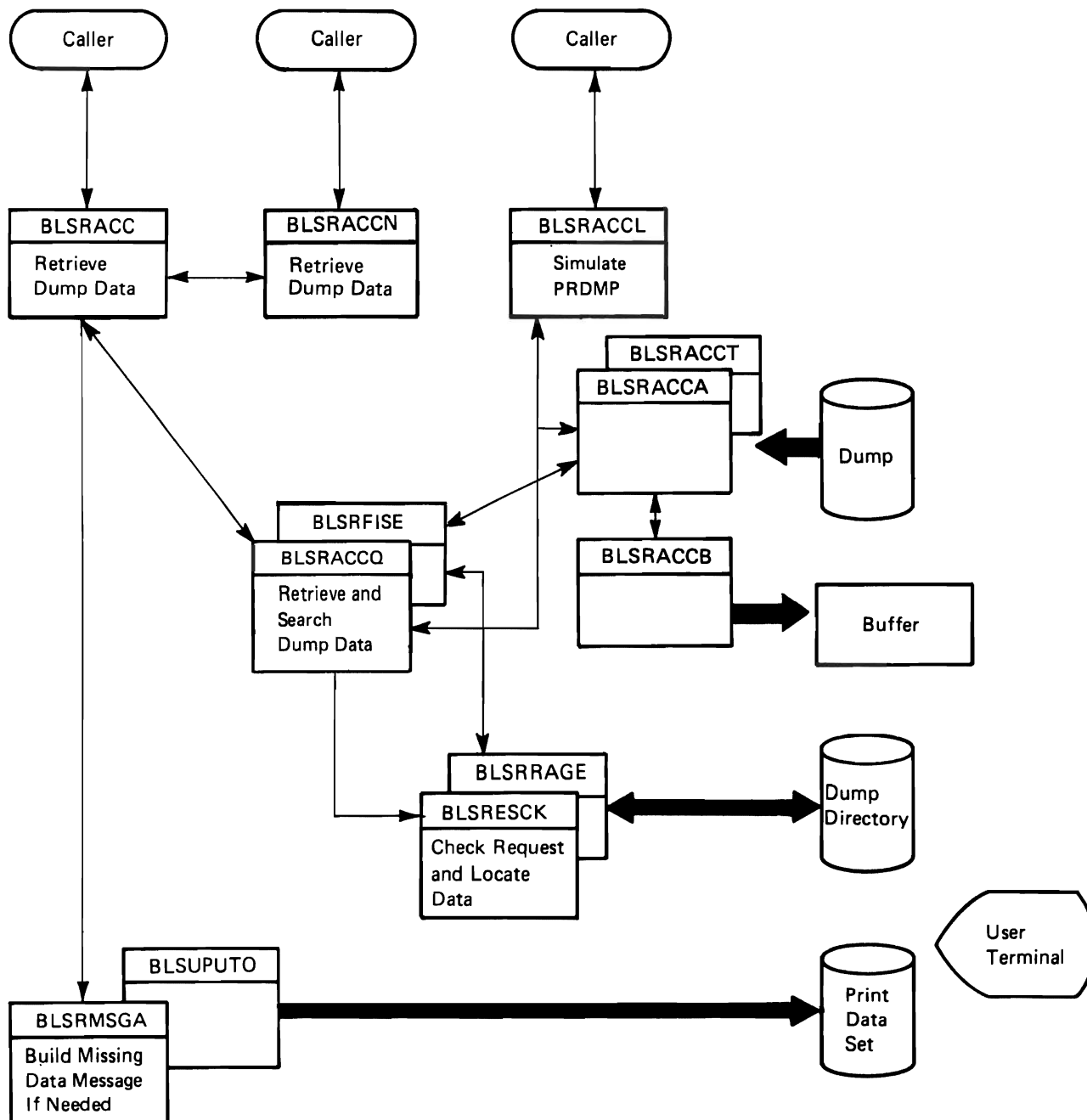


Figure 34. Dump Access

Dump access service routines (BLSRACCx) handle most direct accesses to dump data sets. A pool of dump record buffers is employed to satisfy a significant portion of requests from virtual storage buffers rather than requesting the transfer

of 4104 bytes from a dump data set to virtual storage. Buffer management data is maintained in a compact area, separate from the buffer pool, to minimize page faults.

A special dump access service routine, **BLSRFISE**, is provided to search dump storage on behalf of the **FIND** subcommand (**BLSRFIND**), the **FIND** primary command (**BLSLFISE**) of the dump browse dialog, and the **FIND** primary command (**BLSLFIND**) of the dump display reporter.

Dump Initialization

Dump initialization service routines (**BLSRDUxx**) are employed to determine whether the information required to process the current **IPCS** data set as a dump is available when it is required. If the information is not available, it is established by dump initialization:

- If the dump directory already describes the dump data set, a dump file control block is constructed and a buffer pool obtained.
- If the dump directory does not describe the dump data set, all records in the dump are read and a description of the dump is placed in the dump directory. Then the initialization described above is performed.

A dump initialization service routine, **BLSRDUDR**, is provided to erase the description of a dump data set from the directory when a dump is to be deleted from the dump directory.

Dump initialization processing is entered whenever an **IPCS** user requests access to a dump data set. Dump initialization is always entered at entry point **BLSRDUCK**, a program packaged in load module **BLSUINI2**.

Open Dump

BLSRDUCK determines whether the dump data set is already open. If it is open, an immediate return is made. Otherwise, control is passed to entry point **BLSRDUAL** in nonresident load module **BLSRDUAL**, and the results of **BLSRDUAL** processing are used to determine whether dump-related processing can continue.

Closed Dump

BLSRDUAL determines whether:

- A default data set has been specified.
- The data set can be allocated.
- The data set has direct, partitioned, sequential, or unknown organization.

If any of these conditions cannot be satisfied, the situation is diagnosed and the dump-related processing is terminated. Otherwise, processing continues.

The DSCB for the data set is read (OBTAIN macro). If the block size is zero in the DSCB, track capacity for the device on which the data set resides (DEVTYPE macro) is supplied. A dump table is selected using the DSCB and the data set name, and that table is loaded. Figure 35 describes the criteria used to select the dump table and lists the initialization programs named in each table. See “BLSRDUT” on page 371 for a detailed description of the dump table.

Table	Criteria for Selecting the Table	Programs
BLSRDUT1	For an MVS/XA dump. An MVS/XA dump has: • Direct, sequential, or unknown organization. • Fixed-length, 4104-byte records or records whose record format and block size is not known.	BLSRDUIN BLSRDUOP BLSRDUS1
BLSRDUT2	For a data set which has: • Direct or sequential organization. • Fixed-length (or RECFM=FBS) records whose block size is other than 4104-bytes.	BLSRDUI2 BLSRDU02
BLSRDUT3	For a data set which has: • Direct or sequential organization. • Records whose format is other than fixed.	BLSRDUI3 BLSRDU03
BLSRDUT4	For a partitioned data set directory. A partitioned data set directory is indicated by entering the name of a partitioned data set without a member name.	BLSRDUI3 BLSRDU04
BLSRDUT5	For a member of a partitioned data set. A member of a partitioned data set is indicated by entering the name of a partitioned data set including a member name.	BLSRDUI BLSRDU05

Figure 35. Summary of Dump Tables

The descriptions of the various initialization programs follow.

Sequential Initialization

A sequential initialization routine is invoked for a dump in two circumstances:

- The dump directory contains no records describing the dump.
- The dump directory indicates that sequential initialization for the dump was partially completed at some earlier time. In this case those records describing the dump are deleted (BLSRDUDR), and sequential initialization is restarted.

BLSRDUIN: opens the dump for QSAM locate mode input, then commences to look at each record in the dump data set. If the dump contains no usable data, dump processing is terminated. Otherwise, a description of the dump contents is stored in the dump directory, the dump is closed, the buffer pool acquired by QSAM is freed, and dump processing is permitted to continue.

During BLSRDUIN processing, the IPCS address space mapping module (BLSRRAAR) is called repeatedly to record the contents of each header record, CPU status record, and storage record contained in the dump data set. Work file records, if any, are ignored. In addition BLSRDUIN passes control to the following modules:

BLSRDUIH processes each header record. It establishes the preferred ASID for the dump data set.

BLSRDUIC processes each CPU status record. It establishes the preferred CPU for the dump data set and determines whether the dump describes the status of multiple CPUs.

BLSRDUIS processes each storage record. It processes any summary dump logical records, informing BLSRRAAR that additional blocks of storage are portrayed in these records.

BLSRDUI2: opens the dump for QSAM locate mode input, then commences to count the records in the dump data set. If the dump contains no data, dump processing is terminated. Otherwise, the number of blocks in the dump is stored in the dump directory, the dump is closed, the buffer pool acquired by QSAM is freed, and dump processing is permitted to continue.

BLSRDUI3: calls the open routine named in the dump table to open the dump for BSAM input, then commences to locate each block in the dump data set. If the dump contains no data, dump processing is terminated. Otherwise, the address of each dump block is stored in the dump directory in two records:

ID	Purpose
RB	Associate block numbers with the relative track addresses needed to access dump blocks. See “BLSRRBSY” on page 387 for more information.
RC	Enumerate the relative track addresses of the blocks in the dump. See “BLSRRCSY” on page 388 for more information.

Dump processing is permitted to continue after the data set has been read. The data set is not closed; BSAM will be used throughout for processing data sets initialized by BLSRDUI3.

Open Processing

An open routine is invoked for a dump in two circumstances:

- The dump directory indicates that sequential initialization for the dump was completed at some earlier time. In this case the open routine is invoked immediately.
- The open routine is invoked after successful completion of sequential initialization.

All open routines open the dump data set for input and allocate a pool of buffers (BLSRDUBF). Their distinctive characteristics are summarized in Figure 36.

Module	Access Method	DCB Attributes
BLSRDUOP	BDAM	MACRF=(RIC),OPTCD=R,RECFM=F,BLKSIZE=4104
BLSRDUO2	BDAM	MACRF=(RIC),OPTCD=R,RECFM=F
BLSRDUO3	BSAM	MACRF=(RP),RECFM=U,NCP=1
BLSRDUO4	BSAM	MACRF=(RP),RECFM=F,NCP=1,BLKSIZE=256,LRECL=256,KEYLEN=8
BLSRDUO5	BPAM	MACRF=(R),RECFM=U,NCP=1

Figure 36. Dump Open Routine Summary

Direct Initialization

An direct initialization routine is invoked after successful completion of the open processing unless direct initialization has been performed previously. A direct initialization routine is only supplied for MVS/XA dumps.

BLSRDUS1 locates the communications vector table, the private area and the extended private area for all MVS/XA dumps. For MVS/SA stand-alone dumps, BLSRDUS1 also locates the PSA for the default CPU and the master segment table.

Storage Map Management

The storage map management function examines control blocks for other IPCS functions. For example, if an IPCS find routine has located what is supposed to be a certain control block, a storage map management routine is called to examine the control block and determine whether it appears to be valid. This function builds storage address (SA) records for the validated control blocks. The collection of these records forms a storage map. The availability of the storage map means that IPCS does not have to examine the specific control block more than once. Figure 37 on page 113 depicts the operation of the storage map management function.

Storage map management routines support the IPCS discipline for data validation. If an IPCS routine discovers the address of some MVS/XA data area, it passes the address of the data area to the storage map management routines. Storage map management checks the validity of the data area (if possible).

The storage map consists of a variable number of storage address records. Each storage address (SA) record describes the address and type of a named data area: **STRUCTURE(xxx)**, **AREA(yyy)**, or **MODULE(zzz)**. For certain types it also includes the results of validation on the data area.

The map serves two purposes. It shortcuts the validation process by saving the results of a validation so that it does not have to be redone, and it maps the locations of various MVS/XA data areas for the user.

The process of validation may actually locate new areas. These new areas are added to the map and are also subject to validation. It is possible in this manner to locate most of the major data areas in an MVS/XA dump given a starting point.

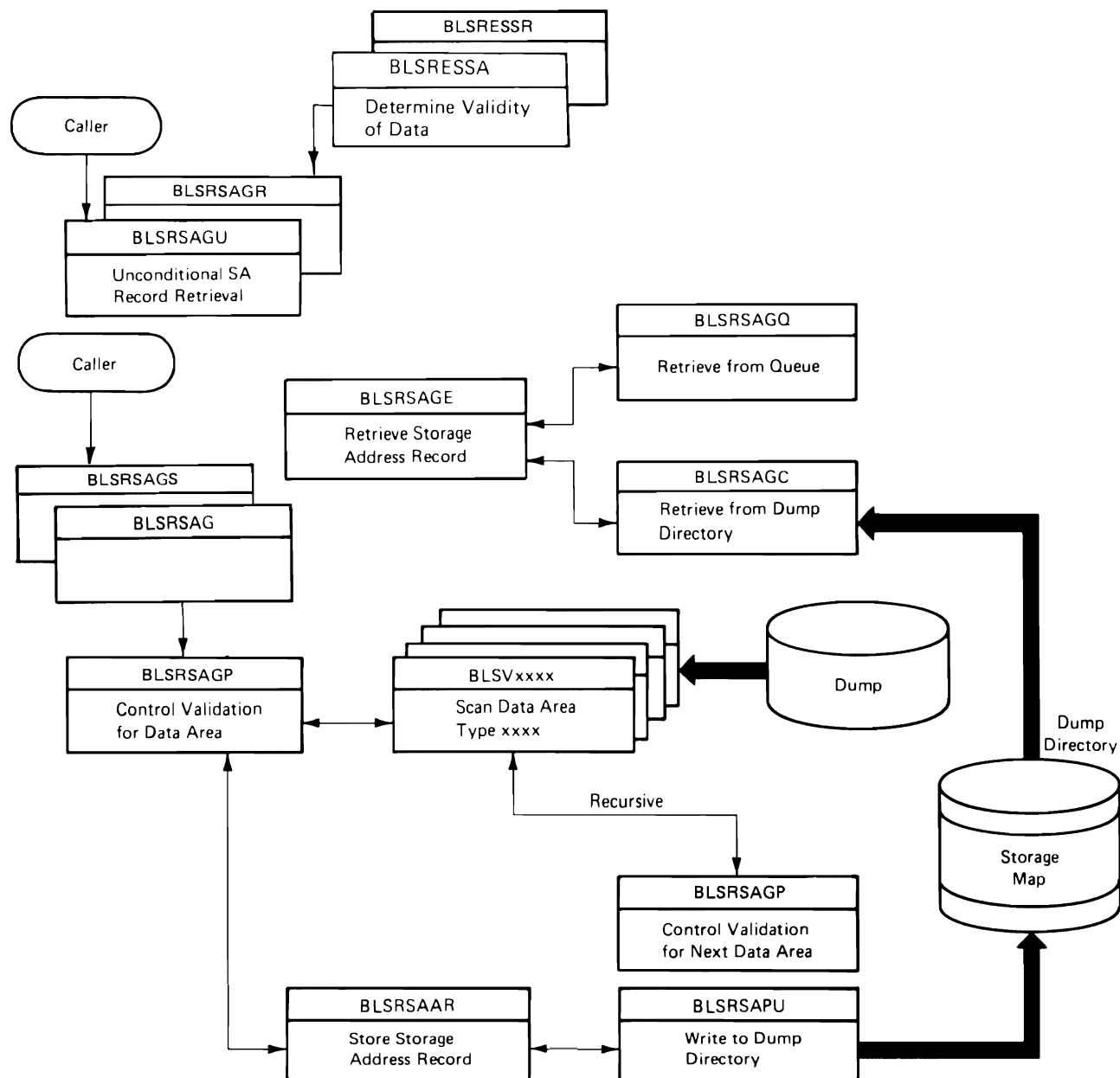


Figure 37. Storage Map Management

The Validation Process

The validation process is controlled by the storage map management routine BLSRSAGP. BLSRSAGP is passed a parameter, a storage address (SA) record, which asserts that a data area of a particular type is located at a certain address. The purpose of the routine is to check the validity of this assertion where possible. An additional parameter may identify how the area was located, whether by a find routine (an equate symbol record) or by a scan routine (an SA record).

If validation results are available on the active storage address record queue (BLSRSAGQ), those results are provided.

If a storage address record exists in the dump directory and complete validation of the block has been performed, no additional work is performed.

If no validation results are recorded in the dump directory and IPCS provides no validation for the specified kind of data, a record is generated indicating that validation has completed and no errors were found. BLSRSAPU is used to add the record to the dump directory.

If no storage address record exists in the dump directory and IPCS provides validation for the specified kind of data, validation of the block is initiated unless a maximum indirection level (DEPTH) has been reached beyond data essential to subcommand processing. Validation is performed by a scan routine (BLSVxxxx) coded to inspect either one kind of data area or any one of a group of closely related kinds of data areas. BLSRSAPU is used to add the resulting record to the dump directory.

If a storage address record exists in the dump directory and validation of the block has not been completed, validation of the block is initiated (or restarted) unless a maximum indirection level (DEPTH) has been reached. The same routine (BLSVxxxx) that initiated validation is invoked as often as required (recursively) to complete its processing. That routine employs the variable-data portion of the storage address record to checkpoint the validation process. If any new validation is completed, BLSRSAAR is used to upgrade the storage address record in the dump directory.

Scan Routines

Scan routines validate significant blocks of data in a dumped system. Each scan routine (BLSVxxxx) consolidates validation logic for a class of control blocks (as indicated by xxxx).

A Hypothetical Scan Routine

Each scan routine is designed to validity check one specific kind of data in as efficient and effective a manner as possible. Complex data types require complex scan routines; however, all scan routines share a common structure which can be best illustrated by a hypothetical scan routine, BLSVWIDG.

For example, BLSVWIDG could validate a group of structures called WIDGETS. There are three types of WIDGETs, all of which share a common architecture. The types are: WIDGETSMALL, WIDGETMEDIUM, and WIDGETLARGE.

- All WIDGETs are aligned on a doubleword boundary.
- All WIDGETs reside in the private area.

- All WIDGETs have a field that is set to CL8'WIDGET'.
- All WIDGETs have a field that contains the length of the WIDGET. The length is 16 bytes for WIDGETSMALL, 64 for WIDGETMEDIUM, and 2K for WIDGETLARGE. No other length is valid.
- All WIDGETs contain pointers to other data areas.

This is the information that is used to validate a WIDGET.

Phases of Scan Routine Execution

The validation process is controlled by the storage map management routines described in a separate section.

There are five phases of scan routine execution. The phases are: pre-fetch analysis, fetch, post-fetch analysis, locator analysis, and message transmission. The results of each phase of execution are saved in the storage address record. Execution may be suspended in the locator analysis phase and resume at a later time. Once a data area has been completely validated the scan routine diagnostic messages may be retransmitted without revalidating the data area.

When BLSVWIDG receives control it is passed a storage address record that specifies the address of the data area. The type of data area must be STRUCTURE(WIDGET). The record also describes the status of the validation process:

- If validation has been completed, this is a request to redisplay diagnostic messages. Such a request is issued only by BLSRSALI (LISTMAP subcommand doing RESCAN processing) or by BLSRSAGU.
- If validation is just starting or is suspended in locator analysis, this is a request to resume validation of the data area. The scan routine was invoked by BLSRSAG.

Pre-fetch Analysis: In the pre-fetch analysis phase any analysis is done that can be done prior to, or that is necessary for, fetch phase processing. This phase is skipped if the scan routine is entered to redisplay diagnostic messages, or to resume suspended locator analysis.

BLSVWIDG checks the boundary alignment and the location of the supposed WIDGET in this phase. The length of the WIDGET is retrieved from the dump and validated for use during the fetch phase. The type of WIDGET is determined from the length. Both the length and type name of the WIDGET are recorded in the SA record.

Fetch Phase: The fetch phase retrieves the entire data area image from the dump. This phase is entered in the normal sequence of validity checking. It is also entered if locator analysis is being resumed, or if messages are to be displayed which require data from the dump.

BLSVWIDG retrieves the image of the WIDGET from the dump using the length that was determined in the pre-fetch phase. This phase tests that the storage is in the dump.

Post-fetch Analysis: In the post-fetch analysis phase, tests are made on the image of the control block. This phase is bypassed if suspended locator analysis is being resumed, or if BLSVWIDG is being executed only to issue messages. BLSVWIDG checks that the WIDGET ID has the value CL8'WIDGET'.

Locator Analysis Phase: The locator analysis phase is the last analysis phase. In this phase pointers are used to locate new data areas. The results of validating these new areas are incorporated in the validation of the current data area. Validation of the current data area is not complete until validation for the daughter control blocks has proceeded at least as far as the post-fetch phase. If the maximum DEPTH has been reached this does not occur and validation is said to be suspended. It may be resumed at a subsequent execution of the scan routine.

In the locator analysis phase BLSVWIDG exploits the pointers in the WIDGET to locate new data areas. It recursively calls BLSRSAG passing it the address and data type of each area located by the WIDGET. SEVERE errors in the located control blocks are severity ERROR for the WIDGET. If BLSRSAG does not validate a new control block because the maximum DEPTH has been reached, locator analysis is suspended.

Message Transmission: The message transmission phase is responsible for informing the user of any significant (according to the FLAG setting) errors discovered during validation. It is isolated from the actual testing so that messages may be reissued without revalidating the WIDGET. The results of all the testing have been saved in the SA record, and it is only necessary to examine certain bits to diagnose the errors that were discovered.

Types of Tests

There is a message associated with each test done by BLSVWIDG. Each message is assigned a severity (WARNING, ERROR, SEVERE). The assignment is heuristic. A severity of SEVERE precludes any possibility that the WIDGET is valid. Boundary misalignment, incorrect WIDGET ID, and invalid length are all SEVERE errors. A severity ERROR indicates something is definitely wrong with the control block; for example, WIDGETLARGE does not point to a valid WIDGETMEDIUM. A severity WARNING indicates something of special interest which is not a definite error, such as "storage not in dump."

In addition, the messages may be broken up into two classes: those that stop validity checking when they occur and those that don't. Most messages that stop validity checking are SEVERE, but this is not a necessity: "Storage not in dump" is a WARNING condition that stops validity checking.

Figure 112 on page 281 relates the types of data validated by IPCS to the modules performing the validation.

Symbol Management

The symbol management function generates equate symbol records and stores them in the symbol table. The operation of the symbol management function is shown in Figure 38 on page 117.



An equate symbol record is used to name a contiguous block of data, such as a control block.

Each equate symbol (ES) record generated and used by IPCS is the result of a search of a dump data set to locate one contiguous block of storage which has unique significance in the dumped system. For example:

CVT	is a symbol that defines the communications vector table in that system.
ASCB001	is a symbol that defines the address space control block for ASID(1) in the system.
PRIVATE	is a symbol that defines the private area in the system.

Equate symbol record management routines support the IPCS discipline for significant data location.

Finding Data Areas

The IPCS modules that find data areas are referred to as find routines. Find routines are used to locate significant blocks of data in a dumped system when access to that data is demanded.

The Find Process

The find process is controlled by BLSRESGE, a symbol management routine.

The find process begins with the symbol table in the dump directory. If a symbol is defined in the symbol table, that definition is provided to the caller.

When the definition of a block of data is not found in the symbol table, the dump table (see “BLSRDUT” on page 371) is used to determine whether a find routine may be used to dynamically resolve the definition:

- If three conditions are satisfied:
 - the data sought is a named area
 - the name of the area appears in the data type processing table (see “BLSRDUTDT” on page 370) addressed by BLSRDUT field DUTAREAP
 - a find routine is identified for the named area

the find routine is loaded and given control to locate the requested area.

- If the data sought is a load module and BLSRDUT field DUTDSL contains the name of a find routine, the find routine is loaded and given control to locate the requested module.
- If three conditions are satisfied:
 - the data sought is a named structure
 - the name of the structure appears in the data type processing table (see “BLSRDUTDT” on page 370) addressed by BLSRDUT field DUTSTRCP
 - a find routine is identified for the named structure

the find routine is loaded and given control to locate the requested structure.

If a find routine is successful, BLSRESGE adds the definition of the symbol to the symbol table in the dump directory as well as providing the definition to its caller.

If a find routine is not successful and the definition must be available to continue a process, the condition is diagnosed and the process is terminated. In this instance, the IPCS user may use the EQUATE subcommand to provide a definition of the key symbol and may reissue the subcommand.

Operation of a Hypothetical Find Routine

Each find routine is designed to locate one specific kind of data in as efficient and effective a manner as possible. If the algorithm required to locate one kind of data is complicated, that complexity is reflected in the find routine. All find routines can be viewed as variations of the find routine for a hypothetical control block, the WIDGET.

Each MVS/XA ASID contains one WIDGET associated with each of its tasks. The address of the WIDGET associated with the task whose task control block (TCB) is named TCB00057AAADC can be calculated by adding the contents of TCBWIDG to the contents of ASXBWIDG. ASXBWIDG is a field in the address space extension block (ASXB) whose name is ASXB00057. The name of the WIDGET associated with TCB00057AAADC is WIDG00057AAADC and the IPCS find routine is named BLSSWIDG.

Examining the Symbol: BLSSWIDG receives control from BLSRESGE whenever both of the following conditions hold:

- The symbol whose definition is to be resolved is not defined in the dump directory.
- The data type specified by the caller of BLSRESGE is STRUCTURE(WIDGET).

BLSRWIDG first examines the symbol to assure that:

- Characters 1-4 of the symbol are "WIDG."
- Characters 5-7 of the symbol are numeric.
- Characters 8-9 of the symbol are alphabetic.
- Characters 10-31 of the symbol are blank.

If the symbol is not properly formed, a return code 12 is returned by BLSSWIDG to indicate that the WIDGET could not be located using the designated input.

Searching the Dumped System: Field ASXBWIDG in ASXB00057 must be retrieved to calculate the address of WIDG00057AAADC. This requires two routine calls:

- BLSRESGU is called to retrieve the definition of ASXB00057, a STRUCTURE(ASXB), and to diagnose a serious condition if the definition cannot be resolved. If ASXB00057 cannot be defined, a return code 12 is returned by BLSSWIDG to indicate that the WIDGET could not be located.
- BLSRACC is called to retrieve the contents of field ASXBWIDG from ASXB00057 and to diagnose a serious condition if the storage cannot be retrieved. If ASXBWIDG cannot be retrieved, a return code 12 is returned by BLSSWIDG to indicate that the WIDGET could not be located.

Field TCBWIDG in TCB00057AAADC must also be retrieved to calculate the address of WIDG00057AAADC. This also requires two routine calls:

- BLSRESGU is called to retrieve the definition of TCB00057AAADC, a STRUCTURE(TCB), and to diagnose a serious condition if the definition cannot be resolved. If TCB00057AAADC cannot be defined, a return code 12 is returned by BLSSWIDG to indicate that the WIDGET could not be located.

- BLSRACC is called to retrieve the contents of field TCBWIDG from TCB00057AAADC and to diagnose a serious condition if the storage cannot be retrieved. If TCBWIDG cannot be retrieved, a return code 12 is returned by BLSSWIDG to indicate that the WIDGET could not be located.

Determining Validity: The contents of ASXBWIDG and TCBWIDG are summed to determine the address of the WIDGET in the Private Area of ASID(57). BLSRSAGU (see the storage map management function) is called to establish an entry for a WIDGET in the map of system blocks and to demand that validation results be returned. If validation cannot be completed successfully or if a serious error is detected in the addressed WIDGET, a return code 12 is returned by BLSSWIDG to indicate that the WIDGET could not be located.

Returning the Description: If validation completes successfully and no serious error is detected in the WIDGET, a return code of 0, 4, or 8 is returned by BLSSWIDG to indicate that no unusual conditions were detected, a warning condition was detected, or an error condition was detected, respectively. The description of the WIDGET supplied by validation processing also is returned showing:

- The number of bytes (signed) between the address of the WIDGET and the first physical byte that it occupies.
- The number of physical bytes that the WIDGET occupies.
- The WIDGET should be viewed as a SCALAR object rather than an array. Neither dimension nor initial array subscript have been provided.
- This WIDGET is a STRUCTURE(WIDGETSMALL). (WIDGET is actually a description of a group of control blocks which share a common architecture. A field common to all forms of WIDGETs permits programs like the validity-check routine for WIDGET to distinguish between a WIDGETSMALL, a WIDGETMEDIUM, and a WIDGETLARGE.)

A brief remark is supplied which describes the function of STRUCTURE(WIDGETSMALL).

Notes:

1. If only one WIDGET is possible in a system, the find routine does not examine the symbol. IPCS routines that request the definition of this kind of block use a single, common reference symbol. But the use of that symbol is not enforced, so that the IPCS user may employ any symbol desired when referencing the block.
2. Although the preceding example does not illustrate the point, several find routines implement multiple search algorithms, selecting one or another based on the characteristics of the dump data set, failure of a preferred algorithm, or other factors.

See Figure 119 on page 298 for more information regarding find routines.

Address Space Mapping

The address space mapping function collects address space records while processing dump data and stores them in a map. The operation of this function is shown in Figure 39 on page 122.



Address space (RA) records link data addresses from an address space to the dump data set records which contain an image of the storage at those addresses in the address space. They answer the question, “What dump data set record contains an image of location 5000?” The answer may be provided in two forms:

- The 50th record in the dump contains the storage of interest. Furthermore, if it is known, the 51st through the 58th records contain the storage immediately following location 5000 in the address space.
- Locations 3000 through 7FFF are not available in this dump.

Address space records are generated during dump initialization and during the processing of the dump. Address space mapping service routines (BLSRRAXx) are employed to manipulate dump directory records which associate storage from an address space in a dumped system with records contained in a dump data set. There are two sources for this information:

- Dump initialization records a basic description of dump data set contents as each record is read from the dump.
- Two routines build on the existing description of the dump. BLSRRACR creates a description of the real storage seen by a MVS/XA processor based on the description of absolute storage generated by AMDSADMP plus the prefix register value obtained from a CPU status record in the dump. BLSRRACV creates a description of MVS/XA virtual storage seen from the perspective of one address space:
 - Common area storage requested for ASID(1), for example, is located, if dumped, in the address space dumped by SDUMP.
 - Segment tables and page tables for the designated address space are consulted to simulate dynamic address translation.
 - If appropriate, processing continues beyond page table entries which have been marked as invalid. The page frame table (PFT) is consulted to simulate the reclaim function performed by MVS/XA real storage management.

Address Resolution

The address resolution function, as shown in Figure 40 on page 124, creates symbol records representing addresses based on user-supplied operands, including indirect addresses. The function examines the parameter descriptor elements (PDEs) for the subcommand, and creates symbol records from them. The symbol records of resolved addresses are used by the subcommand and any other service functions it calls.

The address resolution function consists of nine routines with a naming convention of BLSRADDx. There are two primary entry points into this group of routines: BLSRADDR and BLSRADDS. The invocation of BLSRADDS includes all of the function of BLSRADDR, plus the fact that the symbol X value is updated to the current address value.

As the information is processed, certain absurd conditions are diagnosed by these routines and appropriate error messages are issued to the user. Upon the successful



Dump Directory Handling

As shown in Figure 41 on page 126, the dump directory is a special-purpose VSAM data base, employed by IPCS in much the same manner as a translation-lookaside-buffer is employed by an MVS/XA central processor. As information from the dump is demanded, IPCS anticipates that much of the same information will be demanded again. Records are placed in the dump directory in a manner which will speed second and subsequent references to the data in those records. The dump directory therefore, is a collection of dump data records grouped according to type of data. Wherever possible, the format of this information is adapted to the production of a report useful to a debugger.

A single access control block (ACB) is employed for all IPCS accesses to the dump directory. It is addressed by field ZZ1ACBP in the IPCS Common Area (BLSUZZ1), and is opened during IPCS initialization (see BLSUVSOP and BLSUVSOQ), and closed during IPCS termination (see BLSUVSCL).

The dump directory handling routines perform various VSAM operations on the dump directory data set. They are invoked for the initialization of the data set, by any routine that requires access to the data set, and at IPCS termination time.

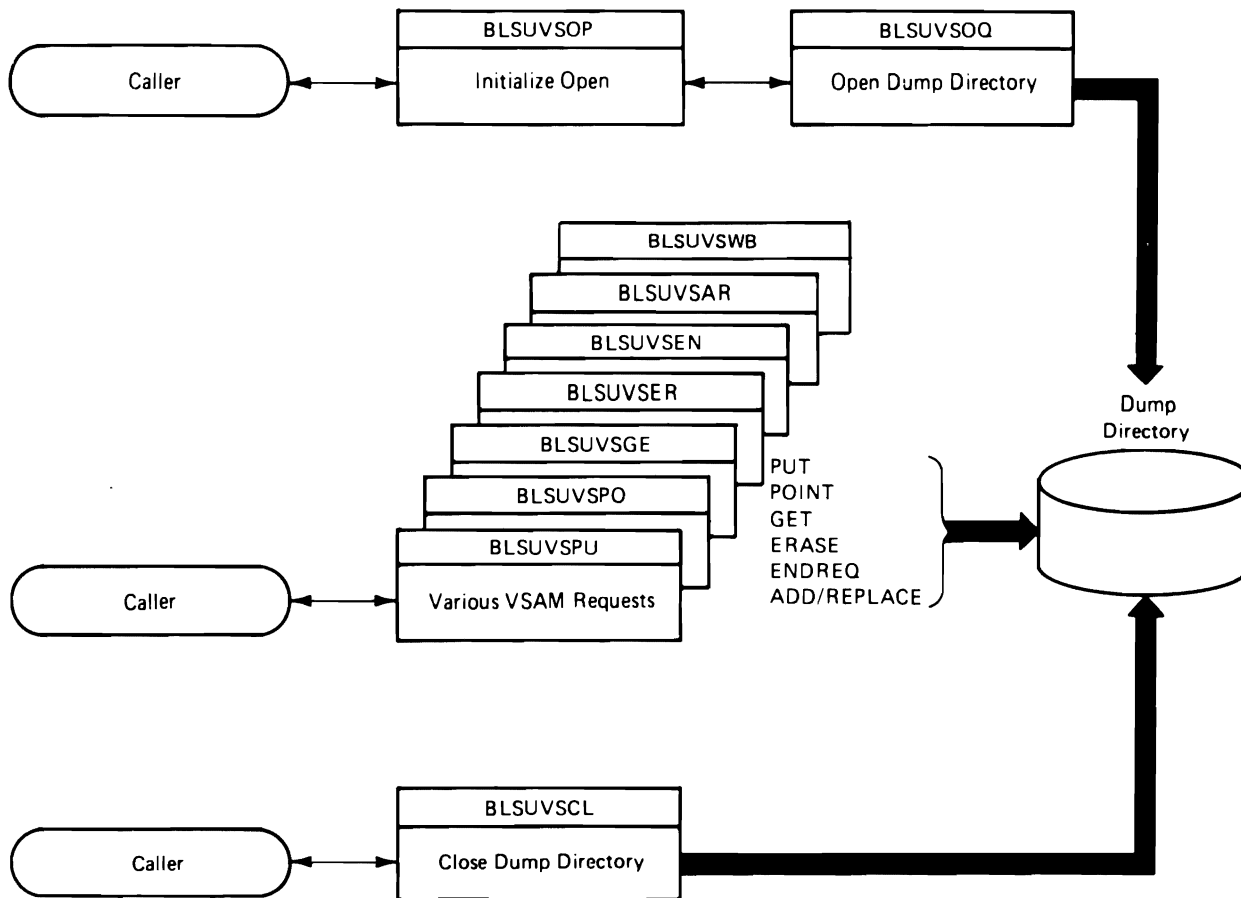


Figure 41. Dump Directory Handling

Initialization and Termination

The BLSUVSOP routine is called by the BLSUINI1 initialization function of IPCS. BLSUVSOP ensures that the dump directory data set has been allocated and that it is a VSAM organization. BLSUVSOQ is called to perform the actual OPEN function. BLSUVSCL is called from BLSU and BLSUSTAE to close the data set at normal and abnormal termination time, respectively.

Accessing the Dump Directory

Twelve different routines provide an interface to various VSAM services. Each routine provides a unique VSAM service as described below. The input to each routine is a pre-initialized request parameter list (RPL). In addition, the BLSUVSCR routine may be invoked to create an RPL and the BLSUVSMR routine may be used to modify an existing RPL.

Dump Directory VSAM Request Routines

All accesses to the dump directory are processed through a group of dump directory VSAM request routines which submit the requests to VSAM for synchronous processing. The routines in this group are usually associated with a single VSAM macro-instruction.

Each dump directory VSAM interface routine requests one or more VSAM services (usually just one), checks return codes, and, if an error has been detected, immediately diagnoses the error. In most cases the VSAMFAIL service routine, IKJEFF19, is employed to assure consistency with other TSO applications.

Dump Directory Record Management Routines

Each IPCS record management routine provides processing associated with one kind of dump directory record, processing required repeatedly in IPCS subcommands. Each such routine has a name of the form BLSRrrff where rr indicates the record type and ff indicates the operation(s) to be performed.

All dump directory accesses required by the record management routines are processed through routines in the VSAM interface group.

Dump Directory Records

Records in the dump directory are maintained in a logical sequence by VSAM. The initial two characters of each record indicate the type of information that it contains and cause it to be placed near other records of the same type in the logical sequence:

ID	Description
X'0000'	The low record consists of 128-bytes of binary zeros. The initial and final records in the dump directory are written by the TSO utility command IPCSDDIR and permit: • IPCS to open the dump directory for update processing. (VSAM does not permit an empty cluster to be opened for update processing.) • IPCS subroutines that access the cluster in sequential mode, or use a search for a key greater than or equal to a value, to consider an end-of-file condition as a logical error.
C'ES'	Equate symbol records associate symbols with contiguous blocks of storage within an address space from a dumped system. The equate symbol records that pertain to one dump collate together in the dump directory, and, taken together, form an alphabetized symbol table for the dump. See "BLSRESSY" on page 373 for more information about equate symbol records.
C'PH'	Program history records are employed by MVS/XA dump initialization and the DSPL3270 subcommand to remember data associated with a dump between sessions. See "BLSRPHSY" on page 381 for more information about program history records.
C'RA'	Address space records link addresses from the address spaces in a dump to the dump data set records which contain an image of the related storage. The address space records that pertain to one dump collate together in the dump directory. They are only used internally by IPCS and are not used directly to compose any report. See "BLSRRASY" on page 385 for more information about address space records.
C'RB'	Block list records link relative block numbers to the relative track addresses used to access data sets whose blocks vary in length. The block list records that pertain to one dump collate together in the dump directory. They are only used internally by IPCS and are not used directly to compose any report. See "BLSRRBSY" on page 387 for more information about block list records.
C'RC'	Contents list records list the relative track addresses of all blocks in dump data sets whose blocks vary in length. The contents list records that pertain to one dump collate together

in the dump directory. They are only used internally by IPCS and are not used directly to compose any report. See "BLSRRCSY" on page 388 for more information about contents list records.

- C'RD' Data set records record the fact that some information regarding the data sets is present in the dump directory. The data set records collate in order by data set name. See "BLSRRDSY" on page 389 for more information about data set records.
- C'SA' Storage address records record the results of validity-checking operations performed on AREA, MODULE, and STRUCTURE data. The records for each dump collate together in the dump directory and, taken together, may be viewed as a map of the control blocks reputed to be in the dumped system. See "BLSRSASY" on page 391 for more information about storage address records.
- C'SD' Session defaults records are employed by the SETDEF (and related IPCS functions) and FIND subcommands to remember data between sessions. See "BLSRSDSY" on page 394 for more information about session defaults records.
- X'FFFF' The high record consists of 128-bytes of binary ones. See the preceding discussion of the low record for more information about the high record.

Properties of the Dump Directory

The dump directory is a VSAM data set having the properties:

CLUSTER

A VSAM cluster contains records which are sorted using a key. These records may be:

- Inserted directly. A user may define symbol B in a symbol table where symbols A and C are already present. When the symbol has been defined, the symbol table will contain A, B, and C in alphabetic order.
- Retrieved directly. The definition of B may be accessed directly.
- Retrieved sequentially. The symbols may be accessed alphabetically.
- Manipulated sequentially. All symbols beginning with A can be deleted or redefined.

KEY(128,0)

The records are sorted using the initial 128 bytes, and no record shorter than 128 bytes may be written. If the application desires (IPCS does), the final portion of the key may be used for the storage of data, provided that less than 128 bytes are necessary to uniquely identify a record. VSAM supports GENERIC key requests where an application specifies that only the initial portion of the key need be matched.

RECORDSIZE(256,3072)

Records vary in length as needed to perform their function. No record longer than 3072 bytes will be stored.

Data Type Keywords	Data Type	Line Caption	Data Format or Formats	Maximum Data Per Line
BIT	B	address	hexadecimal	16 or 32 bytes
POINTER	A	address	hexadecimal	16 or 32 bytes
CHARACTER	C	address	character	64 bytes
SIGNED	F	address	signed decimal	16 bytes
MODULE STRUCTURE	L	offset and address	hexadecimal and character	16 or 32 bytes
Register Address Notation	A C F	Register Number	hexadecimal or character or signed decimal or unsigned decimal	16 bytes
AREA	U	address	hexadecimal and character	16 or 32 bytes
UNSIGNED	Y	address	unsigned decimal	8 or 16 bytes

Figure 42. Data Types and Output Formats

Dump Data Formatting

Dump data formatting service routines (BLSTxxxx) are employed in the display of dumped storage. Available data plus user preferences expressed via the subcommand keywords control the amount of information displayed:

- If the DISPLAY(SYMBOL) or DISPLAY(REMARK) options are in effect and the specified data is available, it is displayed.
- If the DISPLAY(REQUEST) option is in effect or must be forced, a model LIST subcommand is displayed describing the storage.
- If the DISPLAY(MACHINE) option is in effect and a storage key or an underlying absolute address is available, the information is displayed.
- If the DISPLAY(STORAGE) option is in effect, the storage is displayed using a formatting technique determined by the type of data contained in the storage. Missing sections of requested storage are summarized.

The formatting routines are invoked by IPCS subcommands and service routines that need to display data. Usually the data is from the dump being analyzed, but any data can be provided to these routines. The routines are grouped into three categories:

- Interface and control flow routines.
- Data type formatting routines.
- Header message routines.

Interface and Control Flow Routines

As shown in Figure 43 on page 130, two different entry points are provided for calling routines.

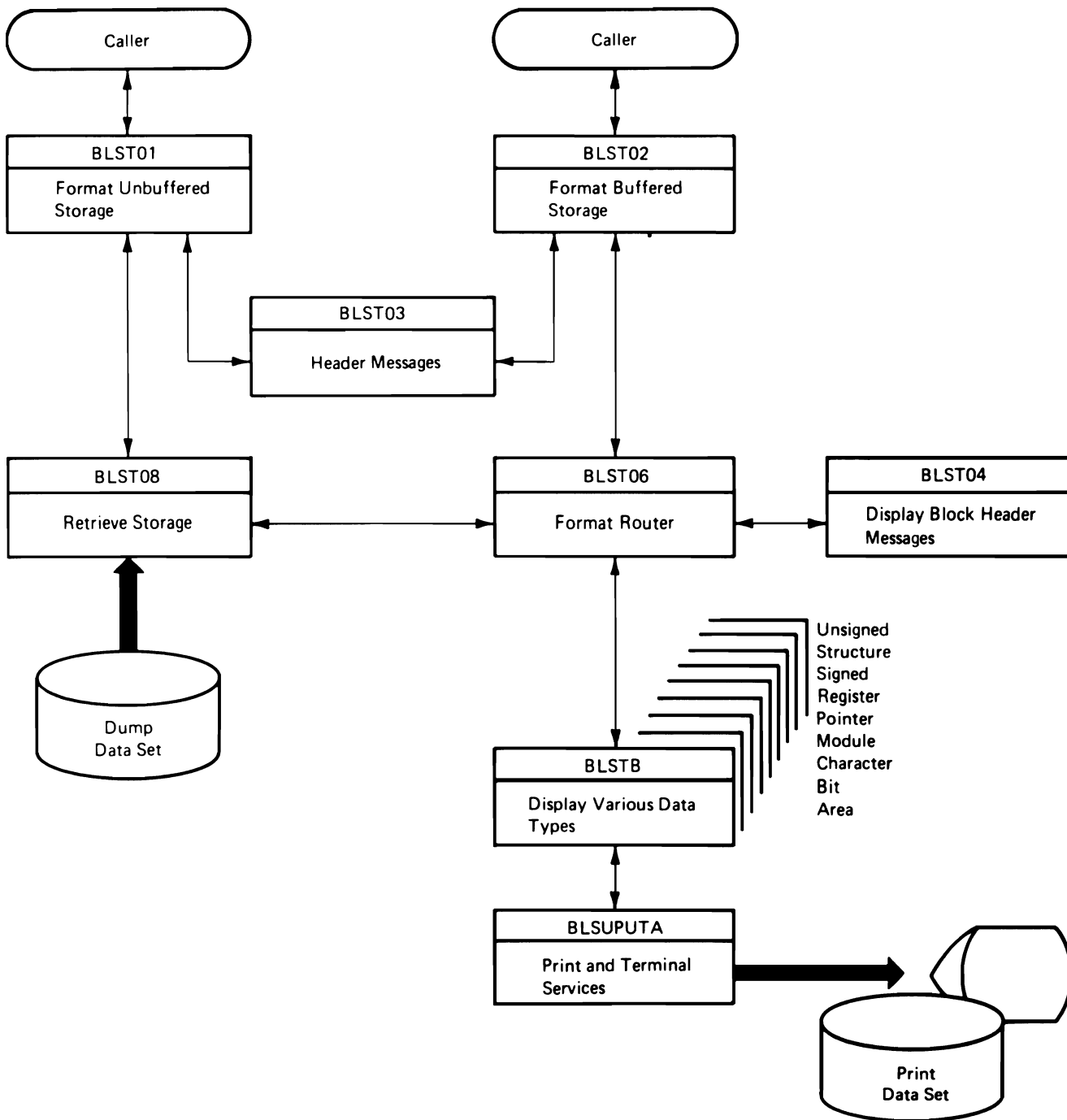


Figure 43. Dump Data Formatting

The BLST01 routine is called if the data to be formatted is not yet in a storage buffer. The BLST02 routine is called if the data to be formatted already exists in a storage buffer. In both cases, a symbol record is required on input that describes the data which is to be formatted. The BLST08 routine serves as an interface to the dump access method by calling the BLSRACCQ routine to retrieve the dump data, if necessary. The format router routine, BLST06, is always called to invoke a particular formatter routine depending on the data type. Each data type is a single unique character located in the symbol record.

Data Type Formatting Routines

The data type formatting routines receive control from the format router, BLST06. Each routine is designed to process a particular data type. If two different data types are designed to have the same output format, then a common routine processes both data types via use of an alias on the main entry point.

Each routine is called by the name BLSTx, where x is the same character as the data type. Figure 42 on page 129 shows the various data types supported and the relationship to the external syntax. Three of the data types (B, M, and U) call different routines depending on the width of the output medium. For each of these three routines, two other routines are provided to transmit either 16 or 32 data bytes per line in the required format. These routines are named BLSTd4 and BLSTd5 where “d” represents the data type and the 4 and 5 indicates 16 and 32 data bytes per line, respectively.

The SIGNED and UNSIGNED data types use the BLST05 routine to handle the various data widths possible and the necessary conversions to decimal.

The BLSTR routine is designed particularly to handle the general and floating-point register references. This routine also recognizes if control register data is being accessed from the HEADER record or the STATUS record and indicates that this control register data is in the output format. General register or control register data can be displayed as hexadecimal, character, signed, or unsigned. Floating-point register data can be displayed only with a data type of bit.

All of the data type routines invoke the BLSUPUTA routines to transmit the formatted data to the currently active output destinations (terminal and/or printer).

Section 3. Program Organization

This section is divided into the same five basic functions as Section 2, namely: initialization and termination, monitor, subcommands, ISPF dialog programs, and common service routines.

Non-Executable Modules

IPCS includes several non-executable, compiler-generated modules that consist of structures. Because the compiler does not guarantee the placement of structures in the output modules, IPCS defines the structures that are to be accessed by other modules as alias entry points. The “primary” entry point of such a module is not directly referenced by other IPCS modules.

The Modifiable Module

All non-executable modules except for module BLSUPUT consist of literal data. BLSUPUT consists of both literal and modifiable data, and each active IPCS user has a separate copy of BLSUPUT.

BLSUPUT: Basic IPCS Data Areas (alias BLSUZZ1)

Function: BLSUPUT is a separate load module containing initialized data areas, some of which are treated as literal data and some of which are modified during the IPCS session:

Modifiable Data

Label	Description
ZZ1	IPCS common area (alias BLSUZZ1). See “BLSUZZ1” on page 404 for more information.
ZZ2	IPCS master task variable. See “BLSUZZ2” on page 406 for more information.

Literal Data

Label	Description
LOCO	Low dump directory record image—128 bytes that contain X'00'.
ESCO	Dump directory equate symbol record image. See “BLSRESSY” on page 373 for more information.
PHCO	Dump directory program history record image. See “BLSRPHSY” on page 381 for more information.
RACO	Dump directory address space record image. See “BLSRRASY” on page 385 for more information.
RBCO	Dump directory block list record image. See “BLSRRBSY” on page 387 for more information.
RCCO	Dump directory contents list record image. See “BLSRRCSY” on page 388 for more information.
RDCO	Dump directory data set record image. See “BLSRRDSY” on page 389 for more information.
SACO	Dump directory storage address record image. See “BLSRSASY” on page 391 for more information.

SDCO Dump directory system default record image. See “BLSRSDSY” on page 394 for more information.

FFCO High dump directory record image—128 bytes that contain X'FF'.

BLSR327W DSPL3270 initial control area image

Read-Only Modules

BLSDVECT: Central Data Base Services Vector Table (alias BLSDVT)

BLSDVECT (alias BLSDVT) contains the DVT. The DVT provides addresses of routines and data vital to the IPCS functions concerned with access to a central data base. See “BLSDVT” on page 343 for more information.

BLSRDTM1: MVS/XA Dump Data Area Table

BLSRDTM1 contains the list of MVS/XA data areas for which find routines or validation routines are present. See “BLSRDTDT” on page 370 for information about table format.

BLSRDTU1: MVS/XA Dump Area Table

BLSRDTU1 contains the list of MVS/XA areas (such as the common and private areas) for which find routines or validation routines are present. See “BLSRDTDT” on page 370 for information about table format.

BLSRDUT1: MVS/XA Dump Table

BLSRDUT1 contains the names and addresses IPCS uses to provide processing for MVS/XA dumps. See “BLSRDUT” on page 371 for information about table format.

BLSRDUT2: RECFM=F Dump Table

BLSRDUT2 contains the names and addresses IPCS uses to provide processing for RECFM=F dumps. See “BLSRDUT” on page 371 for information about table format.

BLSRDUT3: RECFM=U Dump Table

BLSRDUT3 contains the names and addresses IPCS uses to provide processing for RECFM=U dumps. See “BLSRDUT” on page 371 for information about table format.

BLSRDUT4: PDS Directory Dump Table

BLSRDUT4 contains the names and addresses IPCS uses to provide processing for PDS directories processed as dumps. See “BLSRDUT” on page 371 for information about table format.

BLSRDUT5: PDS Member Dump Table

BLSRDUT5 contains the names and addresses IPCS uses to provide processing for PDS members processed as dumps. See “BLSRDUT” on page 371 for information about table format.

BLSRSST1: MVS/XA Dump Special Symbol Table

BLSRSST1 contains the list of special symbols IPCS supports for MVS/XA dumps. See “BLSRSST” on page 395 for information about table format.

BLSRVECT: Dump Analysis Vector Table (alias BLSRVTV)

BLSRVECT (alias BLSRVTV) contains the RVT. The RVT provides addresses of routines and data that the IPCS dump analysis and formatting functions require. See “BLSRVTV” on page 397 for more information.

BLSRVEC2: Dump Analysis Vector Table 2 (alias BLSRV2)

BLSRVEC2 (alias BLSRV2) contains the RV2. The RV2 provides addresses of routines and data that the IPCS dump analysis and formatting functions require.

BLSUSTBL: IPCS Subcommand Table

BLSUSTBL contains the list of IPCS subcommands and the entry points associated with those subcommands.

BLSUVECT: Central Service Vector Table (alias BLSUVT)

BLSUVECT (alias BLSUVT) contains the BVT. The BVT provides the addresses of routines and data required to provide those services shared by all IPCS components. See “BLSBVT” on page 333 for more information.

BLS9PUT: TSO-Mode Control Blocks

BLS9PUT contains initialized data areas used during the processing of the TSO subcommand.

BLS9STBL: TSO-Mode Subcommand Table

BLS9STBL is a subcommand table used during the processing of the IPCS TSO subcommand when it is entered with no command text.

BLS9STB1: TSO-Mode Subcommand Table

BLS9STB1 is a subcommand table used during the processing of the IPCS TSO subcommand when it is entered with command text.

Initialization and Termination

The program control flow showing the modules involved in the initialization and termination process appears in Figure 44 .

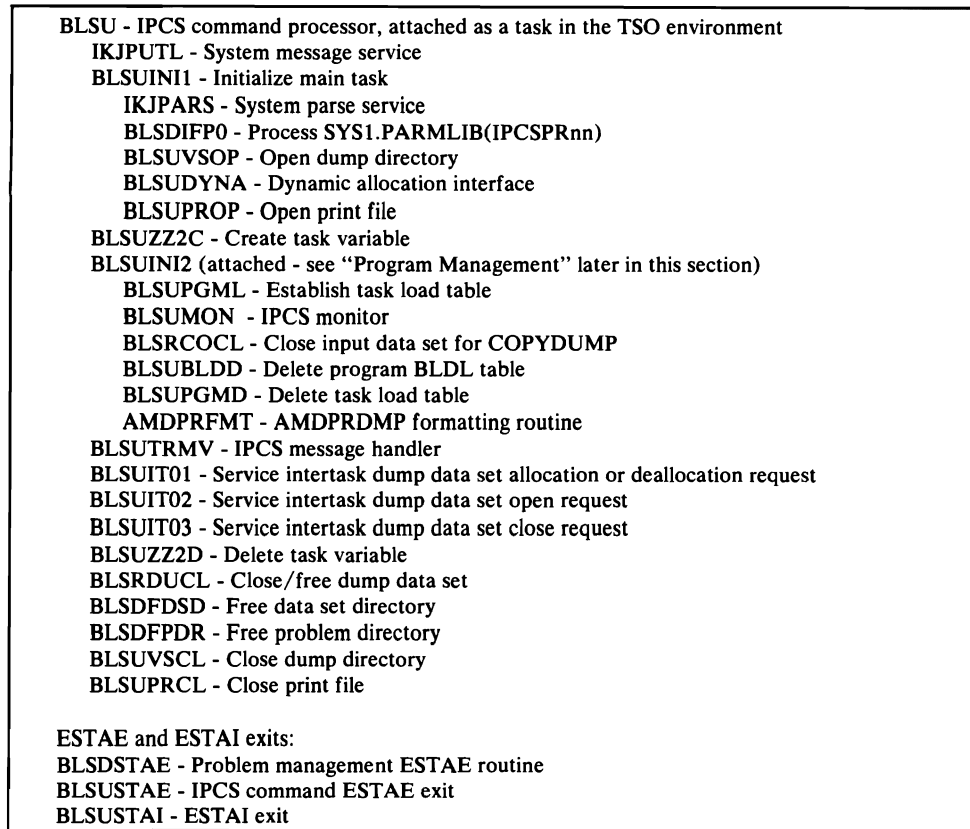


Figure 44. Initialization and Termination Control Flow

For lower levels of control flow (for example, calls to common service routines), use the following individual module descriptions.

BLSDIFP0: Process SYS1.PARMLIB(IPCSPRnn)

Function: BLSDIFP0 allocates SYS1.PARMLIB(IPCSPRnn), LINKs to BLSDIFP1 to read the member, and unallocates SYS1.PARMLIB(IPCSPRnn).

Modules Called: BLSDIFP1, BLSUALLO, BLSUDYNA, BLSUFRE1

Input

1. Master task variable (see "BLSUZZ2" on page 406).
2. The decimal digits to be appended to 'IPCSPR' to form the name of the active IPCS SYS1.PARMLIB member.

Output: No values are returned directly to the caller. However, the IPCS facility parameter block (FPBLOK) is allocated and initialized, and PARMLIB dependent fields in the IPCS control block BLSUZZ1 are established.

Data Areas Used: CPPL, ECT, IOPL, UPT

BLSDIFP1: Read SYS1.PARMLIB(IPCSPRnn)

Function: This module reads the IPCS SYS1.PARMLIB member IPCSPRnn where each n represents one decimal digit. The processing consists of allocating, opening, and reading the member, performing validity checks on the member contents, establishing IPCS system values, closing the member, and deallocating it.

Modules Called: BLSADSD, BLSADPDR, BLSUALLO, BLSUFRE1, BLSUTRMV, IKJGETL, IKJPARS, IKJSTCK

Input

1. Master task variable (see “BLSUZZ2” on page 406).
2. The decimal digits to be appended to ‘IPCSPR’ to form the name of the active IPCS SYS1.PARMLIB member.
3. The DDNAME allocated to SYS1.PARMLIB.

Output: No values are returned directly to the caller. However, the IPCS facility parameter block (FPBLOK) is allocated and initialized, and PARMLIB dependent fields in the IPCS control block BLSUZZ1 are established.

Data Areas Used: CPPL, ECT, IOPL, UPT

BLSDINI0: Setup BLSDB Environment

Function: Initialize IPCS central DB support environment

Modules Called: BLSCALOC, BLSADSD, BLSADPDR, BLSDENQ0, BLSDFPDR, BLSDBSGS, BLSDBMDG0

Input: The active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSRZZ3R: Refresh Defaults

Function: Alter the IPCS session defaults.

Modules Called: BLSRSDGE, BLSUALLO, BLSUFRE1

Input: The active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSU (alias IPCS): IPCS Command Processor

Function: This module is the entry point to IPCS.

1. BLSU loads entry point BLSUZZ1, an entry point associated with an initialized image of an IPCS common area
2. BLSU calls BLSUINI0 which loads those TSO service routines which are required during the IPCS session

3. BLSU places the entry points of service routines in the master IPCS task variable, an initialized image of which was retrieved with BLSUZZ1.
4. BLSU establishes BLSUSTAE as the IPCS STAE exit for the master task.
5. BLSU invokes BLSUINI1 via LINK so that those functions performed only once during an IPCS session can be performed, and the storage used can be made available for reuse during the remainder of the session.
6. BLSU calls BLSUMOAT to ATTACH and supervise the IPCS processing task.
7. BLSU calls a sequence of clean-up routines to release the system resources acquired during the session
8. BLSU returns to its caller.

Modules Called: BLSDFPDR, BLSDINI0, BLSUINI0, BLSUINI1, BLSUINI2, BLSUINI9, BLSUMOAT, BLSUPRCL, BLSUTLCL, BLSUVSCL, IKJPUTL

Input: Command processor parameter list (CPPL).

Output: No data is returned to the caller.

Data Areas Used: CPPL

BLSUINI0: Setup IPCS Environment

Function: Initialize the IPCS environment.

Modules Called: BLSUALLO, BLSUDYNA, BLSUFRE1, BLSUMPKN, BLSUMPK1, BLSUPROP, BLSUPRTA, BLSUPRTN, BLSUPRTT, BLSUPUTN, BLSUSTAI, BLSUSTDE, BLSUSTKS, BLSUTRMA, BLSUTRMN, BLSUTRMV, BLSUVSCA, BLSUZZ0, BLSUZZ0T, BLSUZZ2C, BLSUZZ2D, IKJEFF02, IKJEFF18, IKJEFF19, IKJEFT25, IKJGETL, IKJPARS, IKJPTGT, IKJSCAN, IKJSTCK

Input: The active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSUINI1: Initialize IPCS Command Processing

Function: This module performs the following processing to initialize the IPCS command:

1. Parses the command operands.
2. Reads the active IPCS SYS1.PARMLIB member (IPCSPRNN).
3. Opens the dump directory data set (IPCSDDIR).
4. Opens the print file (IPCSPRNT).

Modules Called: BLSDIFP0, BLSUALLO, BLSUFRE1, BLSUTLCK, BLSUTLOP, BLSUVSOP, IKJPARS.

Input: The active task variable (see “BLSUZZ2” on page 406)

Output: No values are returned directly to the caller.

Data Areas Used: ECT, UPT

BLSUINI2: IPCS Dump Processing Task Monitor

Function: This module receives control from BLSU after IPCS data sets have been allocated and the processing task variable has been created. It sets several fields in the processing task variable and then establishes the task load table (BLSUPGML). This table is an internal record of which IPCS modules have been loaded into storage. If a non-terminating return code is received, BLSUMON (the first monitor routine) is called, and returns control only when the IPCS session ends.

Modules Called: BLSRZZ3R, BLSUALLO, BLSUFRE1, BLSUINI3, BLSUINI8, BLSUMON, BLSUMONI

Input: None

Output: A task-load table

Data Areas Used: CPPL, PSCB, ECT, UPT, PSA

BLSUINI3: IPCS Processing Setup

Function: Prepare the environment for an IPCS processing task or ISPF logical screen task.

Modules Called: BLSDESTAE, BLSRSTAE, BLSUALLO, BLSUFRE1, BLSRUPGML

Input: The active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSUINI8: Processing Task Cleanup

Function: BLSUINI8 frees resources acquired by an IPCS processing task or an ISPF logical screen task.

Modules Called: BLSDFPDR, BLSDESTAE, BLSQEXTI, BLSRCOCL, BLSRDUCL, BLSRSTAE, BLSUALLO, BLSUBLDD, BLSUFRE1, BLSUPGMD

Input: The active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSUDYNA: Request Dynamic Allocation

Function: This module requests dynamic allocation services by means of the DYNALLOC macro. If an error is detected, it diagnoses the error using the DAIRFAIL service routine, IKJEFF18.

Modules Called: BLSUALLO, BLSUFRE1, IKJEFF18

Input

1. Master task variable (see “BLSUZZ2” on page 406).
2. Dynamic allocation (SVC 99) parameter list, completely initialized.

Output: The dynamic allocation parameter list and the data addressed by it may be updated by SVC 99 to reflect the actions performed.

BLSUMOAT: Attach IPCS Task

Function: Create a task variable and attach an IPCS task using that variable. Provide standard inter-task services during the execution of the task. If the task terminates abnormally, reinstate the task unless there is a possibility that this would produce a loop.

BLSUINI2 is invoked via ATTACH from BLSUMOAT to process IPCS subcommands until the end of the session. During BLSUINI2 processing BLSUMOAT remains sensitive to requests for intertask service, currently required only for print file OPEN, and CLOSE processing.

Module	Service
BLSUIT01	DYNALLOC (SVC 99)
BLSUIT02	OPEN (SVC 19)
BLSUIT03	CLOSE (SVC 20)
BLSUIT04	OPEN print file
BLSUIT05	CLOSE print file

If an ABEND occurs at a point during BLSUINI2 processing when it is reasonable to attempt to continue the session, BLSUMOAT deletes the task under which BLSUINI2 initially ran and reattaches BLSUINI2.

When BLSUINI2 terminates in a way that implies an end of the IPCS session, BLSUMOAT returns to BLSU to complete the IPCS session.

Modules Called: BLSUINI2, BLSUIT01, BLSUIT02, BLSUIT03, BLSUIT04, BLSUIT05, BLSUSTAI, BLSUSTDE, BLSUTRMV, BLSUZZ2C, BLSUZZ2D, BLS9INI2

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. Entry point name for the subtask

Output: None

BLSUPRAB: Print File DCB ABEND Exit

Function: Contains code used for DCB ABEND processing of the print file DCB. Its entry point was provided during generation of the DCB in BLSUPROP.

Modules Called: BLSUTRMV

Input: DCB ABEND exit parameter list

Output: None

BLSUPRCL: Close Print File

Function: This module closes the IPCS print file.

Input: Master task variable (see “BLSUZZ2” on page 406)

Output: None

Data Areas Used: DCB

BLSUPROP: Open Print File

Function: This module opens the IPCS print file. The print file must be preallocated. DCB parameter validation is performed during open exit processing.

Modules Called: BLSUALLO, BLSUFRE1

Input: Master task variable (see “BLSUZZ2” on page 406)

Output: None

Data Areas Used: DCB

BLSUPROX: ICPSRNT DCB Open Exit

Function: Open the DCB for the IPCS print file, using the entry point generated in BLSUPROP.

Input: Print file DCB

Output: None

BLSUZZ2C: Create Task Variable

Function: This module obtains storage for a task variable and initializes that storage.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A fullword

Output: The fullword contains the address of the task variable created.

BLSUZZ2D: Delete Task Variable

Function: This module releases the storage used by a task variable.

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. A fullword containing the address of the task variable to be deleted

Output: None

IPCS STAE Exits

The following modules serve as ESTAE and ESTAI exits for IPCS. Each module's description has the same inputs, outputs, and data areas.

BLSDESTAE: Problem Management and Tracking ESTAE Routine

Function: This module ensures that the problem directory and data set directory are closed and, if allocated under the monitor task, are freed. This ESTAE module does request retry. If an error occurs, the termination of IPCS is requested.

Modules Called:: BLSDFPDR

Input: Standard MVS parameters to an ESTAE (or ESTAI) exit.

Output: None

Data Areas Used: SDWA

BLSRSTAE: Dump Related ESTAE Exit

Function: BLSRSTAE closes and frees IPCS dump data sets.

Modules Called: BLSQEXTI, BLSRCOCL, BLSRDUCL

Input: Standard MVS parameters to an ESTAE (or ESTAI) exit.

Output: None

Data Areas Used: SDWA

BLSUSTAE: IPCS Command ESTAE Exit

Function: BLSUSTAE closes and frees IPCS data sets.

Modules Called:: BLSDFPDR, BLSUPRCL, BLSUTLCL, BLSUVSCL

Input: Standard MVS parameters to an ESTAE (or ESTAI) exit.

Output: None

Data Areas Used: SDWA

BLSUSTAI: ESTAI Exit

Function: BLSUSTAI generates a diagnostic message when a subtask of IPCS terminates abnormally. It suppresses that message and dump processing when the abend is the result of a user-requested detach of a TSO command (including IPCS).

Modules Called: BLSUVSEN, IKJEFF19

Input: Standard MVS parameters to an ESTAE (or ESTAI) exit.

Output: None

Data Areas Used: SDWA

BLSUSTAS: Dialog Task ESTAE Exit

Function: BLSUSTAS removes a dialog task variable from the chain of task variables (see “BLSUZZ2” on page 406) and releases the storage associated with it.

Modules Called: BLSUZZ2D

Input: Standard MVS parameters to an ESTAE (or ESTAI) exit.

Output: None

Data Areas Used: SDWA

BLSUSTDE: Subtask Detach ESTAE Exit

Function: BLSUSTAS detaches all tasks attached by the abending IPCS task, permitting their ESTAE exits to receive control.

Modules Called:: None

Input: Standard MVS parameters to an ESTAE (or ESTAI) exit.

Output: None

Data Areas Used: SDWA

BLS9STAE: TSO Mode ESTAE Exit

Function: This module generates a diagnostic message when a TSO mode task abends.

Modules Called:: BLSVTLCL, IFJEFF19

Input: Standard MVS parameters to an ESTAE (or ESTAI) exit.

Output: None

Data Areas Used: SDWA

Monitor

The monitor module descriptions are grouped into three subfunctions:

- Automatic storage management
- Subcommand monitor
- Program management

Automatic Storage Management

Most automatic storage IPCS uses is managed by two subroutines packaged in module BLSUALLO.¹ These subroutines acquire storage from MVS subpool 1 in 64K blocks and take advantage of the fact that the last automatic storage requested will be the next freed to reduce the path lengths of IPCS modules.

BLSUALLO: Allocate Automatic Storage

Function: BLSUALLO allocates automatic storage to IPCS modules.

Input: All input is passed in registers.

Reg	Contents
0	The number of bytes of storage required. Note: this number should be an integral multiple of 8 bytes to ensure that routines receive storage aligned on doubleword boundaries. No rounding is performed by BLSUALLO.
9	The address of the active task variable (see “BLSUZZ2” on page 406).
13	The address of a 72-byte register save area into which the calling subroutine has already saved all input registers.
14	The address of the return point in the calling program.
15	The address of BLSUALLO.

Output: The address of the storage allocated is returned in register 1.

BLSUFRE1: Release Automatic Storage

Function: BLSUFRE1 is an entry point in BLSUALLO. Its purpose is to release automatic storage.

Input: All input is passed in registers.

Reg	Contents
0	The number of bytes of storage to be freed.
1	The address of the active task variable (see “BLSUZZ2” on page 406).
14	The address of the return point in the calling program.
15	The address of BLSUFREE.

Output: None

¹ Those subroutines that function in STAX or ESTAE exit environments and require automatic storage acquire that storage from MVS directly via GETMAIN.

Subcommand Monitor

The program control flow for the subcommand monitor is shown in Figure 45.

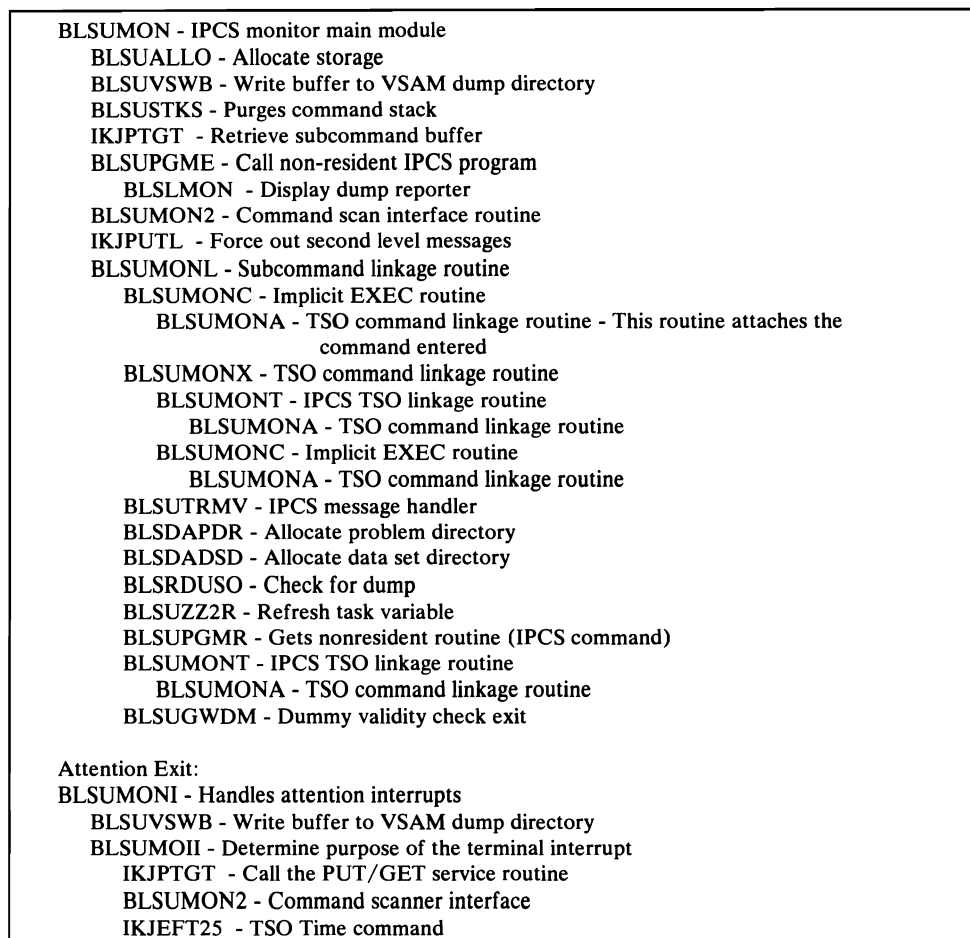


Figure 45. Subcommand Monitor Function Control Flow

For lower levels of control flow, refer to the following module descriptions.

BLSUGWDM: Dummy Validity Check Exit

Function: BLSUGWDM is called during processing of the IPCS TSO subcommand and all TSO commands invoked under IPCS. A BR14 module is supplied with IPCS. An installation command name validation routine that replaces this module may limit access to TSO commands and CLISTs appropriate during an IPCS session.

Input

1. CPPL (macro IKJCPPL)
2. CSOA (macro IKJCSOA)

Output: Bits CSOABAD and CSOAEXEC may be set on.

BLSUMOII: STAX Exit

Function: BLSUMOII establishes a conversation with the TSO user to determine the purpose of a terminal attention interruption. A null response causes the processing suspended by the attention interruption to be resumed. An ABEND or TIME request is processed immediately. Any other subcommand is treated as a request for two actions:

1. Processing that was active when the attention interruption was received is terminated. BLSUMOII effects the termination indirectly unless ABEND was requested. It POSTs field ZZ2EVE in its active task variable. That field is checked frequently during IPCS processing when an orderly shutdown of processing may be initiated.
2. The subcommand is processed.

Modules Called: BLSUMON2, IKJEFT25, IKJPTGT

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. Mode message

Output: None

BLSUMON: Buffer Retrieval Loop

Function: BLSUMON iteratively picks up the command buffer, checks to see if IPCS should terminate, and if not, calls BLSUMONL to process the command.

Before BLSUMON terminates and if the dump display reporter processing is active, BLSUMON calls BLSLMON to present a final view of the dump data report and to release resources acquired by the dump display reporter processing.

Modules Called: BLSLMON, BLSUALLO, BLSUFRE1, BLSUMONL, BLSUMON2, BLSUPGME, BLSUSTKS, BLSUVSWB, IKJPTGP, IKJPUTL

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

Data Areas Used: CSOA, IOPL, PGPB, STPB, CSPL

BLSUMONA: TSO Command Linkage Subroutine

Function: BLSUMONA calls BLSUBLDL to locate a TSO command processor. If one is found, control is passed to it via ATTACH. Field ZZ2CPPL in the active task variable is used for the parameter list.

BLSUMONA completes the processing of a TSO command by deleting its task (DETACH macro) and marking the data sets it used as “not in use” (DYNALLOC macro).

Modules Called: BLSUALLO, BLSUBLDL, BLSUDYNA, BLSUFRE1, BLSUSTAI, BLSUSTDE, BLSUTRMV, BLSUVSWB

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSUMONC: Monitor Implicit EXEC Routine

Function: BLSUMONC indicates that an implicit CLIST request is being processed as an IPCS subcommand:

- A command name (ECTPCMD) of “IPCS” and a subcommand name (ECTSCMD) of “EXEC” are indicated in the ECT (macro IKJECT).
- The second halfword in the command buffer is zeroed.

BLSUMONA is called to pass control to the TSO EXEC command processor. EXEC will either prepare the CLIST for processing or will diagnose any error that blocks successful completion.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMONA

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSUMONI: IPCS Attention Exit

Function: BLSUMONI processes attention interruptions for the IPCS command:

1. BLSUVSWB is called to complete any dump directory updates that are queued when the attention interruption is received.
2. If the DSPL3270 subcommand is active when the attention interruption is received, the TSO terminal I/O controller is informed that line mode processing should be resumed and the terminal screen is cleared.
3. Any terminal input queued when the attention interruption was received is purged using the TCLEARQ macro.
4. A message
IPCS*

is written to indicate that BLSUMONI has received control.
5. BLSUMOII is called to process requests for BLSUMONI.
6. If the DSPL3270 subcommand is to resume processing after BLSUMONI completes, the TSO terminal I/O controller is informed that full-screen mode processing should be resumed.

Modules Called: BLSUMOII, BLSUVSWB

Input

1. TAIE (macro IKJTAIE)
2. Null input indication
3. Active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSUMONL: Subcommand Linkage Subroutine

Function: BLSUMONL categorizes the subcommand based on the results of command scan processing:

1. If extended implicit CLIST notation was used, BLSUMONL passes the subcommand to BLSUMONC for processing as an implicit CLIST.
2. If the verb does not appear in the subcommand table, BLSUMONL passes the subcommand to BLSUMONX for simulation of type 1 terminal monitor program processing.
3. If the verb is described as an unsupported function, BLSUMONL diagnoses the restriction, and simulates completion with a user ABEND code 16.
4. If the verb is described as an IPCS subcommand, BLSUMONL schedules any initialization required prior to passing control to the subcommand (BLSDAPDR, BLSDADSD, BLSRDUSO) and passes control to the subcommand processor via BLSUPGMS.
5. If the verb is described as a TSO command, BLSUMONL passes the subcommand to BLSUMONT for processing as a TSO command.

Modules Called: BLSDADSD, BLSDAPDR, BLSRDUSO, BLSUALLO, BLSUFRE1, BLSUGWDM, BLSUMONC, BLSUMONT, BLSUMONX, BLSUPGMR, BLSUPGMS, BLSUPGMU, BLSUTRMV, BLSUZZ2R

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. CSOA (macro IKJCSOA) that has been completed by IKJSCAN

Output: None

Data Areas Used: CSOA

BLSUMONT: IPCS TSO Linkage Subroutine

Function: BLSUMONT places the TSO command verb in ECT field ECTPCMD and calls BLSUMONA to attach the TSO command processor associated with the command.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMONA

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

Data Areas Used: CPPL, PSCB, ECT, UPT

BLSUMONX: TSO Command Linkage Subroutine

Function: BLSUMONX calls BLSUBLDL to determine whether a TSO command processor is available that has the same name as the verb located in the command buffer by IKJSCAN.

BLSUMONX's processing continues depending on one of the following resultant checks from BLSUBLDL:

- If the command processor is available and the command is the CALL command, the TEST command, or any authorized command listed in IKJEFTE2, BLSUMONX calls IKJEFASR to continue processing.
- If the command processor is available but does not satisfy the criteria for processing by IKJEFASR, BLSUMONX calls BLSUMONT to continue processing.
- If the command processor is not available, BLSUMONX calls BLSUMONC to process the subcommand as an implicit CLIST invocation.

Modules Called: BLSUALLO, BLSUBLDL, BLSUFRE1, BLSUMONC, BLSUMONT, BLSUTRMV, IKJEFASR

Input: Active task variable (see "BLSUZZ2" on page 406)

Output: None

Data Areas Used: CVT, JSCB, LWA (mapped by IKJEFLWA), PSA, TCB, TSVT

BLSUMON2: Command Scan Interface Subroutine

Function: BLSUMON2 calls IKJSCAN to locate the verb in a subcommand buffer. If invalid command syntax or another problem is detected, BLSUMON2 diagnoses the problem.

Modules Called: BLSUALLO, BLSUFRE1, BLSRTRMV, IKJSCAN

Input

1. The active task variable (see "BLSUZZ2" on page 406)
2. CSPL (macro IKJCSPL). Fields CSPLCBUF and CSPLOA must be initialized.

Output: The command scan output area addressed by field CSPLOA and the command buffer addressed by field CSPLCBUF are updated by IKJSCAN.

Data Areas Used: CSOA, CSPL

BLSUSCM1: Process Special Subcommand

Function: BLSUSCM1 processes one subcommand and any additional subcommands implied by that subcommand.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMON, BLSUSTKS, BLS9MO11

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. A fullword containing the length of the subcommand text
3. The subcommand text

Output: None

BLSUSTKD: Delete a Terminal Element

Function: BLSUSTKD deletes a base element from the input stack for a new environment control table (using macro IKJECT).

Modules Called: BLSUALLO, BLSUFRE1, IKJSTCK

Input: The active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSUSTKS: Perform STACK Service

Function: BLSUSTKS passes a request to IKJSTCK and diagnoses any problem encountered.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUTRMV, IKJSTCK

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. STPB (macro IKJSTPB)

Output: None

BLSUSTKT: STACK a Terminal Element

Function: BLSUSTKT adds a base element to the input stack for a new environment control table (using macro IKJECT).

Modules Called: BLSUALLO, BLSUFRE1

Input: The active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSUZZ2R: Refresh Task Variable

Function: BLSUZZ2R restores fields in a task variable to their default values.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A fullword containing the address of the task variable to be refreshed

Output: None

Program Management

The program control flow for the program management function is shown in Figure 46.

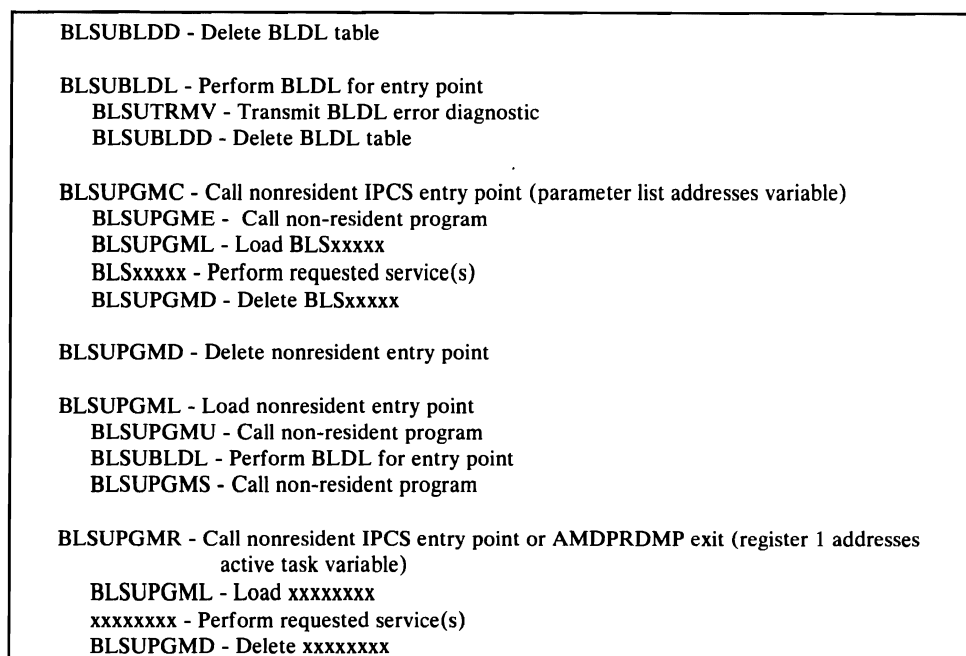


Figure 46. Program Management Control Flow

For lower levels of control flow, refer to the following individual module descriptions.

Program BLDL Table Management

BLSUBLDD: Delete BLDL Table Service Routine

Function: BLSUBLDD releases (via FREEMAIN) an existing BLDL table containing information accumulated by BLSUBLDL.

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSUBLDL: BLDL Data Service Routine

Function: BLSUBLDL returns BLDL information regarding a program entry point to its caller. It builds a table containing the results of all prior BLDL requests and uses that table to provide the requested if it has been requested before.

Modules Called: BLSUALLO, BLSUBLDD, BLSUBLDL, BLSUFRE1, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. BLDL parameter list

Output: None

Non-Resident Module Management

BLSUPGMC: Call Non-Resident IPCS Program

Function: BLSUPGMC loads (BLSUPGML), calls, and deletes (BLSUPGMD) a non-resident IPCS program that expects to receive control with register 1 pointing to a format parameter list in which the first parameter is the active IPCS task variable (see “BLSUZZ2” on page 406).

BLSUPGMC is used rather than BLSUPGME when the absence of the non-resident program should be diagnosed only once during an IPCS session without regard to the number of attempts to use it.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPGML, BLSUPGMD

Input: The active task variable (see “BLSUZZ2” on page 406). ZZ2PGM must contain the name of the program to be called.

Output: None

BLSUPGMD: Delete Non-Resident Entry Point

Function: BLSUPGMD deletes an IPCS program from storage.

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. The 8-character name of the program

Output: None

BLSUPGME: Call Non-Resident Program

Function: BLSUPGME loads (BLSUPGMU), calls, and deletes (BLSUPGMD) a non-resident IPCS program that expects to receive control with register 1 pointing to a format parameter list in which the first parameter is the active IPCS task variable (see “BLSUZZ2” on page 406).

BLSUPGME is used rather than BLSUPGMC when the absence of the non-resident program should be diagnosed each time during an IPCS that an attempt is made to use it.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPGMD, BLSUPGMU

Input: The active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSUPGML: Load Non-Resident Entry Point

Function: BLSUPGML provides the caller with a pointer to a non-resident entry point:

1. A non-resident entry point table is consulted to determine if the requested entry point has already been loaded. If it has, its address is returned to the caller.

BLSUBLDL is called to determine whether the entry point can be accessed in the current environment.

2. If it is only available in the job pack area or the link pack area, the entry point address is provided by BLSUBLDL and returned to the caller.
3. Otherwise, the BLDL results provided by BLSUBLDL are used to speed load processing, and the address provided by load is returned to the caller.

BLSUPGML is used rather than BLSUPGMU when the absence of the non-resident program should be diagnosed only once during an IPCS session without regard to the number of attempts to use it.

Modules Called: BLSUALLO, BLSUBLDL, BLSUFRE1, BLSUTRMV

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. An 8-character entry point name
3. A fullword buffer

Output: The fullword is set to the address of the entry point.

BLSUPGMR: Call Non-Resident IPCS Program

Function: BLSUPGMR loads (BLSUPGML), calls, and deletes (BLSUPGMD) an IPCS program that expects to receive control with register 1 pointing to the active IPCS task variable (see “BLSUZZ2” on page 406).

BLSUPGMR is used rather than BLSUPGMS when the absence of the non-resident program should be diagnosed only once during an IPCS session without regard to the number of attempts to use it.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPGML, BLSUPGMD

Input: The active task variable (see “BLSUZZ2” on page 406).

Output: None

BLSUPGMS: Call Non-Resident Program

Function: BLSUPGMS loads (BLSUPGMU), calls, and deletes (BLSUPGMD) an IPCS program that expects to receive control with register 1 pointing to the active IPCS task variable.

BLSUPGMS is used rather than BLSUPGMR when the absence of the non-resident program should be diagnosed each time during an IPCS session that an attempt is made to use it.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPGMD, BLSUPGMU

Input: The active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSUPGMU: Load Non-Resident Program

Function: Provide the caller with the entry point address of a non-resident entry point.

BLSUPGMU is used rather than BLSUPGML when the absence of the non-resident program should be diagnosed each time during an IPCS session.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUPGML, BLSUTRMV

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. An 8 character entry point name
3. A fullword buffer

Output: The fullword is set to the address of the entry point.

Subcommand Processors

This section contains module descriptions for the modules that execute IPCS commands and subcommands. The section is divided by subcommand type. Each subsection contains the module descriptions in alphabetic order.

The program control flow charts at the front of Section 3 show the module flow used by each subcommand processor. All module names that appear in program control flow charts are listed in the Index. A module may not be described in the same section in which it appears in a chart because a module can appear in more than one chart. Therefore the index should be used to locate a given module description.

A module description appears under the subcommand which uses that module the most. All modules are described only once.

This section is divided as follows:

- General subcommand processing
- Parse validity checking for subcommands
- TSO utility commands
- Session control subcommands
- Problem and data management subcommands
- Analysis subcommands
- PRDMP exit support subcommands
- Line-oriented dump viewing subcommands
- Full-screen dump viewing modules

- CLIST support subcommands
- Dump directory handling subcommands

General Subcommand Processing Modules

The BLSRPADS, BLSUPARU, and BLSUSIGR modules are used by many IPCS subcommands to process display operands, routing operands, and signed integer ranges respectively.

BLSRPADS: Process Display Operands

Function: BLSRPADS translates the PDE(s) generated for display options coded on debugging subcommands to option bits in the active task variable (see “BLSUZZ2” on page 406).

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. The IKJPARS data list (PDL) containing the IKJPARS data elements (PDEs) for the display keywords.
3. An offset table describing the position of each display operand PDE in the PDL.

Output: None

BLSUPARU: Process Routing Operands

Function: BLSUPARU determines the desired message routing from IKJPARS PDEs for the TERM/NOTERM and PRINT/NOPRINT options. It records the message routing in the active task variable (see “BLSUZZ2” on page 406).

Modules Called: BLSUALLO, BLSUFRE1, BLSUPGMR, BLSUPROP, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. The PDL produced by IKJPARS.
3. An offset table which describes the locations of the routing PDEs within the PDL.

Output: None

BLSUSIGR: Process Signed Integer Range

Function: BLSUSIGR translates a string described by an IKJIDENT FIRST=ANY,OTHER=ANY PDE to a pair of fullword signed binary integers.

Modules Called: BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. An IKJIDENT FIRST=ANY,OTHER=ANY PDE
3. An fullword
4. An fullword

Output: The first and second fullwords are set to the signed binary numbers associated with the origin and end of the range described by the PDE.

Parse Validity Checking for Subcommands

IPCS subcommands use the TSO parsing function (IKJPARS) to parse their operands. IKJPARS allows the invoking subcommand to specify a validity check exit for a given PDE.

The exit is entered after IKJPARS has done its own validity checking and all tests have passed. The exit then does its own tests. If the operand appears valid, return code 0 is returned; otherwise, a return code 4 to PARSE generates the message prompt:

```
INVALID xxx  
REENTER -
```

where xxx is text supplied in the parameters passed to IKJPARS.

Some exits return code 8, which tells PARSE to simply produce

```
REENTER -
```

since the exit has produced its own message.

See *TSO Guide to Writing a TMP or a Command Processor* for general information on IKJPARS and validity check exits. The IPCS parse validity checking routines are described below, following the parse interface routine. There are two groups of parse validity checking modules: one for the problem and data management subcommands, and the other for dump related subcommands.

Interface to IKJPARS

BLSUPARI: IKJPARS Interface

Function: This module passes control to IKJPARS. If an unexpected return code is returned, it diagnoses the situation using IKJEFF19.

Modules Called: BLSUALLO, BLSUFRE1, IKJEFF02, IKJEFF19, IKJPARS

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. The address of a parse control list (PCL) that describes the syntax of the operands which may be parsed and specifies the diagnostic text to be displayed should required operands be omitted or should operands be entered improperly.

3. A fullword in which the address of a parameter descriptor list (PDL) is returned.

Output: The address of a PDL that describes the operands which were entered.

IKJPARS Validity Check Routines

The modules listed below are called by IKJPARS to verify that the contents of the PDE conforms to the format and range of values being validated. A return code of 4 is issued when the PDE content does not meet the validation criteria.

Input: A parameter list containing:

1. The address of the PDE built by IKJPARS.
2. The address of the active task variable (see “BLSUZZ2” on page 406).
3. A field initialized to X'FF000000' by IKJPARS.

Output: None

Module	Validation Criteria
BLSDC070	Length is no more than 7 characters.
BLSDC128	Length is no more than 128 characters.
BLSDC600	Length is no more than 60 characters.
BLSDDT00	Date is entered in MM/DD/YY format.
BLSDPC00	Range of problem numbers is ascending.
BLSDSV00	Severity is in the inclusive range from 0 to 4.
BLSDTM00	Time is entered in HH:MM:SS format.
BLSRVPAD	IKJIDENT FIRST=ANY,OTHER=ANY string can be processed by BLSRADGP as an address range syntactically valid for IPCS use.
BLSRVPAS	ASID is in the inclusive range from 1 to 65,534.
BLSRVPCP	CPU address is in the inclusive range from 0 to 15.
BLSRVPVA	IPCS can resolve the value described.
BLSRVPVS	Data set name does not include a member name.
BLSUTLCK	No member name is designated and, if the name is entered without apostrophes and without the final qualifier 'LOAD', it is shorter than 40 characters.
BLSUVPS1	IPCS can resolve the signed integer described.
BLSUVPX1	A valid hexadecimal number is described.
BLSUVPX2	A valid, ascending hexadecimal range is described.
BLSUVP21	Integer is in the inclusive range from 1 to 65,535.
BLSUVP31	Integer is in the inclusive range from 1 through 2 ³¹ .
BLSUVP32	Integer range is ascending and in the inclusive range from 0 through 2 ³¹ -1.
BLSUVP33	Integer is in the inclusive range from 0 through 2 ²⁴ -1.
BLSUVP99	Integer range is ascending and is in the inclusive range from 0 to 99.

TSO Utility Commands

IPCS contains two commands that operate directly under TSO. The commands are used to scan data sets containing dumps, and to initialize an IPCS dump directory data set.

SYSDSCAN Command

BLSRDUSC: IPCS SYSDSCAN Command Processor

Function: BLSRDUSC allocates a given range of SYS1.DUMPnn data sets, opens each, reads up to 10 records searching for a header record, and, if found, displays the date, time, and title of the dump from it. The range of nn may vary within the limits of 00 to 99, with a default range of 00 to 09. The data sets are deallocated after use.

Modules Called: BLSUMTOD, BLSUVP99, IKJEFF18, IKJPARS, IKJPUTL

Input: CPPL (macro IKJCPPL)

Output: None

Data Areas Used: UPT, PSCB, ECT, DCB, CPPL, IOPL, PTPB, IEFZB4D0, IEFZB4D2

IPCSDDIR Command

The program control flow for the IPCSDDIR command is shown in Figure 47 .

BLSRVSLD - IPCSDDIR main processor
IKJPARS - Parse subcommand operands
IKJEFF18 - Issue DAIRFAIL diagnostics
IKJPUTL - Call PUTLINE service routine
IKJEFF19 - Issue VSAMFAIL diagnostics
IKJEFF18 - Issue DAIRFAIL diagnostics

Figure 47. IPCSDDIR Command Control Flow

For lower levels of control flow, refer to the individual module descriptions.

BLSRVSLD: IPCSDDIR Command Processor

Function: This module initializes a VSAM cluster for use as a dump directory. It parses the command, allocates, opens, writes two records (key all X'00' and key all X'FF'), closes, and deallocates the data set. Diagnostic messages are generated by DAIRFAIL and VSAMFAIL.

Modules Called: BLSRVVPS, IKJPARS, IKJPUTL, IKJEFF18, IKJEFF19

Input: CPPL (macro IKJCPPL)

Output: None

Data Areas Used: ACB, RPL

Session Control Subcommands

There are three subcommands in IPCS that help the user establish default values for IPCS parameters, and also permit invocation of TSO commands from the IPCS environment.

HELP Subcommand

BLSUHELP: IPCS HELP Subcommand Processor

Function: This module determines the name of the subcommand for which help is desired.

1. If the subcommand is not identified as an IPCS subcommand, it is assumed that the subcommand is a CLIST for which a separate HELP data set member has been prepared. The ECT is modified to cause the TSO HELP command processor to look for a separate member in the HELP data set.
2. If the subcommand is identified as a TSO command processor, the ECT is modified to cause the TSO HELP command processor to look for a separate member in the HELP data set.
3. If the subcommand is identified as an IPCS subcommand, the ECT is left alone. This causes the TSO HELP command processor to search the IPCS member of the HELP data set for the required HELP information.

The TSO HELP command processor is invoked to actually provide the requested display.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMONA, BLSUPARI

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

Data Areas Used: CPPL

SETDEF Subcommand

The program control flow chart for the SETDEF subcommand is shown in Figure 48 on page 161.

BLSRSETD - IPCS SETDEF main processor
 BLSUALLO - Allocate automatic storage
 BLSUPARI - Parse subcommand operands. Diagnose any error
 BLSUVP31 - Verify ASID, CPU address, data length integers
 BLSUVP33 - Validate integer 0 through $2^{24}-1$
 BLSUPARU - Set message routing options
 BLSRPADS - Set dump data display options
 BLSRDUSE - Set default dump source
 BLSUPGME - Call non-resident program
 BLSRRDGE - Get RD record
 BLSRADD2 - Generate address space information
 BLSRDUCC - Close and free dump data set
 BLSUPGMR - Call non-resident program
 BLSRZZ3U - Update defaults
 BLSUTRMA - Transmit subtitle to the terminal
 BLSUPRTT - Identify (later delete) subtitle for the print file
 BLSRMDSN - Format data set name
 BLSUMPKN - Compress a message
 BLSUMPK1 - Compress a message
 BLSUPGME - Call non-resident program
 BLSRRDGE - Get RD record
 BLSRMSGGA - Format ASID, CPU if appropriate
 BLSUPRTT - Cancel subtitle
 BLSUPUTA - Route LIST option messages
 BLSUFRE1 - Release automatic storage

Figure 48. SETDEF Subcommand Control Flow

For lower levels of control flow, refer to the individual module descriptions.

BLSRSETD: SETDEF Subcommand Processor

Function: The SETDEF subcommand establishes operand defaults for IPCS subcommands. SETDEF also displays operand defaults after any requested changes have been made.

Modules Called: BLSRADD2, BLSRDUCC, BLSRDUSE, BLSRMDSN, BLSRMSGGA, BLSRPADS, BLSRRDGE, BLSRZZ3U, BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUMPK1, BLSUPARI, BLSUPARU, BLSUPGME, BLSUPGMR, BLSUPUTA, BLSUPRTT, BLSUTRMA, BLSUVP31, BLSUVP33

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: Default operand values for subcommands have been established.

BLSRZZ3U: Update Session Defaults

Function: Record the IPCS session defaults.

Modules Called: BLSRSDAR, BLSUALLO, BLSUFRE1, BLSUPGME

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

TSO Subcommand

BLSUTSO: TSO Subcommand Processor

Function: This routine processes the TSO subcommand.

Modules Called: BLSUALLO, BLSUFRE1, BLSUGWDM, BLSUMOAT, BLSUMONT, BLSUMON2, BLSUTRMV, BLS9CMD1, BLS9INI2, IKJEFT25

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

BLS9CMD1: TSO Mode Subcommand Processor

Function: Provide support for execution of one TSO command or CLIST. No input is solicited from the terminal.

Modules Called: BLSUBLDD, BLSUINI0, BLSUINI9, BLSUMON, BLSUSTKS, BLS9MOI1, BLS9STAE, BLS9STB1, IKJPUTL, IKJSTCK

Input: CPPL (macro IKJCPPL)

Output: None

BLS9INI2: TSO Mode Processing Task Monitor

Function: Set various BLSUZZ2 fields and establish the task load table. Call BLSUMON if a non-terminating return code is received. Control will not be returned until the TSO session is ended, or terminates because of an error condition.

Modules Called: BLSUALLO, BLSUBLDD, BLSUFRE1, BLSUMON

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

BLS9MOI1: Special Terminal Attention (STAX) Exit

Function: Support attention interruption of a command or CLIST in a context where only ABEND, END, or TIME requests may be honored. No entry of a command to be executed after termination of the interrupted process is solicited.

There are three situations where BLS9MOI1 is used in IPCS:

- BLS9MOI1 is established as a STAX exit when the TSO subcommand is entered with command text. The command(s) processed as a result of this may be terminated to resume IPCS processing but may not be replaced by another stream of TSO commands through the use of the attention mechanism.
- BLS9MOI1 is established as a STAX exit by the BLSGSCMD dialog processor. The IPCS subcommand(s) processed as a result of this may be terminated to resume ISPF dialog processing but may not be replaced by another stream of IPCS subcommands through the use of the attention mechanism.

- BLS9MOI1 is established as a STAX exit by the BLSLDISP dialog processor when the IPCS primary command is entered. The IPCS subcommand(s) processed as a result of this may be terminated to resume the BLSLDISP dialog but may not be replaced by another stream of IPCS subcommands through the use of the attention mechanism.

Modules Called: BLSUMOII

Input

1. TAIE (macro IKJTAIE)
2. A fullword zero
3. Active task variable (see “BLSUZZ2” on page 406)

Output: None

BLS9MONI: TSO Mode Terminal Attention (STAX) Exit

Function: BLS9MONI supports attention interruption of TSO mode. This is a mode initiated by the entry of the TSO subcommand without command text and terminated through use of the the END subcommand.

Modules Called: BLSUMOII

Input

1. TAIE (macro IKJTAIE)
2. A fullword zero
3. Active task variable (see “BLSUZZ2” on page 406)

Output: None

COPYDUMP Subcommand

The program control flow for the COPYDUMP subcommand is shown in Figure 49.

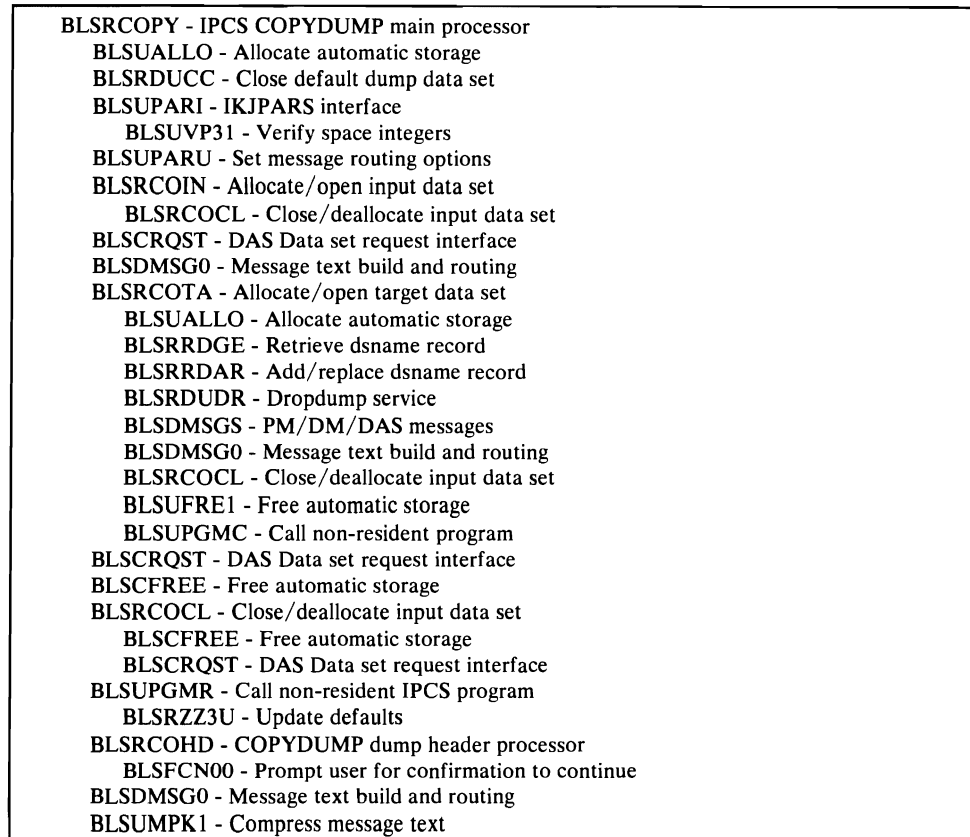


Figure 49. COPYDUMP Subcommand Control Flow

For lower levels of control flow, refer to the individual module descriptions.

BLSRCOPY: COPYDUMP Subcommand Main Processor

Function: This module copies dump records from an input data set into an output data set. The dump title of each dump is displayed as it is encountered during processing. The subcommand allows the user to skip dumps, copy a dump, or simply display all the dump titles in the data set. The input data set as well as the target data set, if not allocated, is dynamically allocated by COPYDUMP. After each copy operation the input data set may be left open or closed depending on the option chosen.

Modules Called: BLSCFREE, BLSCRQST, BLSDMSGSGS, BLSDMSG0, BLSRCOCL, BLSRCOHD, BLSRCOIN, BLSRCOTA, BLSRDUCC, BLSRZZ3U, BLSUALLY, BLSUFRE1, BLSUMPK1, BLSUPARI, BLSUPARU, BLSUPGMR, BLSUVP31

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSRCOCL: Close Input Data Set for COPYDUMP

Function: This module closes the input dump data set used by the COPYDUMP subcommand. It also frees the DCB and the related buffer pools. It then frees the CCB.

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSRCOHD: COPYDUMP Dump Header Processor

Function: This module processes the header record of the dump being processed by the COPYDUMP subcommand. The dump title (if present in the dump) is displayed, and if the CONFIRM option is in effect, the user is prompted as to whether or not the dump is to be copied.

Modules Called: BLSDMSGs, BLSDMSG0, BLSDVT, BLSFCN00, BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUPGMU

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. MVS/XA dump header record

Output: None

BLSRCOIN: Allocate Input Data Set for COPYDUMP

Function: This module first determines whether the input data set was specified via DDNAME or DSNNAME. Then dynamic allocation is performed for the input data set which becomes the source data set for the COPYDUMP subcommand.

Modules Called: BLSCALOC, BLSCRQST, BLSDMSGs, BLSDMSG0, BLSRCOCL, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. DSNNAME of input data set
3. DDNAME of input data set
4. Flag bytes for internal processing

Output: None

BLSRCOTA: Allocate Target Data Set for COPYDUMP

Function: This module dynamically allocates and opens the data set for the COPYDUMP subcommand.

Modules Called: BLSDMSGs, BLSDMSG0, BLSRCOCL, BLSRDUDR, BLSRRDAR, BLSRRDGE, BLSUALLO, BLSUFRE1, BLSUPGMC

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Target data set PDE for DSNAME
3. Target data set PDE for DDNAME
4. Primary space allocation
5. Secondary space allocation
6. Flag bytes for internal processing

Output: None

Problem and Data Management Subcommands

The problem and data management subcommands provide problem and data management functions, adding and deleting, maintaining status, displaying and printing information about problems and their associated data sets.

The program control flow charts for the problem and data management subcommands are presented in Figure 50 through Figure 57 on page 172 .

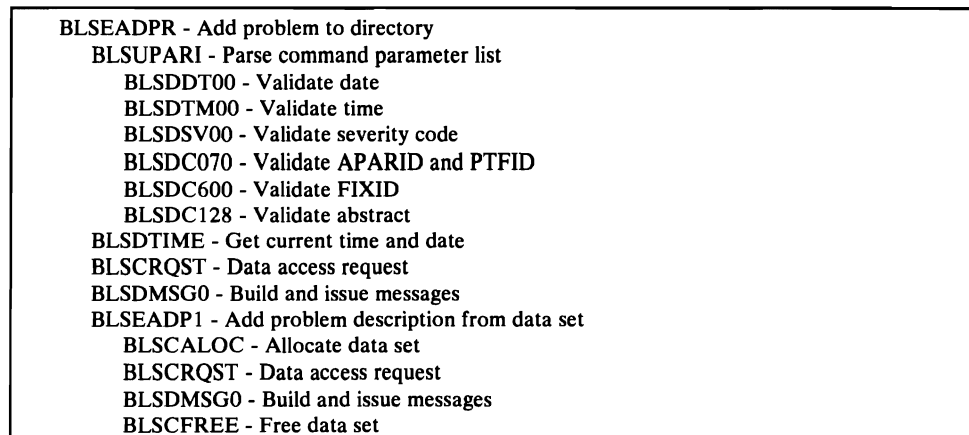


Figure 50. ADDPROB Subcommand Control Flow


```

BLSEDEL0 - Execute DELPROB subcommand
  BLSUPARI - Interface to IKJPARS to parse subcommand
  BLSMSG0 - Message processor
  BLSQST - Enqueue the problem
  BLSRQST - Data access services, open, read, and close PDR
  BLSEAUTH - Check user authorization to delete the problem
BLSEDEL1 - Perform deletion function
  BLSEDEL0 - Handle confirmation option
  BLSRQST - Open and close PDR
  BLSMSG0 - Message processor
  BLSELPCL - LISTPROB controller
  BLSFCN00 - Prompt user for confirmation
  BLSMSG0 - Message processor
  BLSEDDDP - Erase records from problem and data set directories and handle
    disassociation of data sets
  BLSFLALL - Open PDR, build association list, close PDR
  BLSELPFR - Build list of data sets associated with problem
    BLSRQST - Get a record, position a data set
    BLSMSG0 - Issue a message
  BLSFOPEN - Open a data set
  BLSFCLOS - Close a data set
    BLSRQST - Close a data set
  BLSMSG0 - Issue a message
  BLSEFDALL - Delete all problem-data set associations on a list
  BLSMSG0 - Issue a message
  BLSEFDEL - Delete problem-data set associations from database
    BLSFOPEN - Open a data set
    BLSRQST - Get a record
    BLSMSG0 - Issue a message
    BLSFPRES - Erase a PDR record
      BLSFREAS - Erase a record
        BLSRQST - Erase a record
        BLSMSG0 - Issue a message
      BLSMSG0 - Issue a message
    BLSFREAS - Erase a record
  BLSRDUCC - Close and free dump data set
  BLSEFDRES - Erase a record from the DSD
    BLSRQST - Get a record
    BLSFREAS - Erase a record and handle errors
      BLSRQST - Erase a record
      BLSMSG0 - Issue a message
    BLSFCLOS - Close a data set
    BLSFSCRT - Scratch a data set
      BLSALOC - Allocate a data set
      BLSFREE - Free a data set
      BLSRQST - Open a data set, close a data set
      BLSMSG0 - Issue a message
  BLSFPREA - Erase a series of PDR records
    BLSRQST - Get a record
    BLSFREAS - Erase a record and handle errors
      BLSRQST - Erase a record
      BLSMSG0 - Issue a message
  BLSELPFM - Free storage occupied by association list
    BLSMSG0 - Issue a message
  BLSFOPEN - Open a data set
  BLSFCLOS - Close a data set
  BLSMSG0 - Issue a message
  BLSDDDEQP - Dequeue the problem

```

Figure 51. DELPROB Subcommand Control Flow

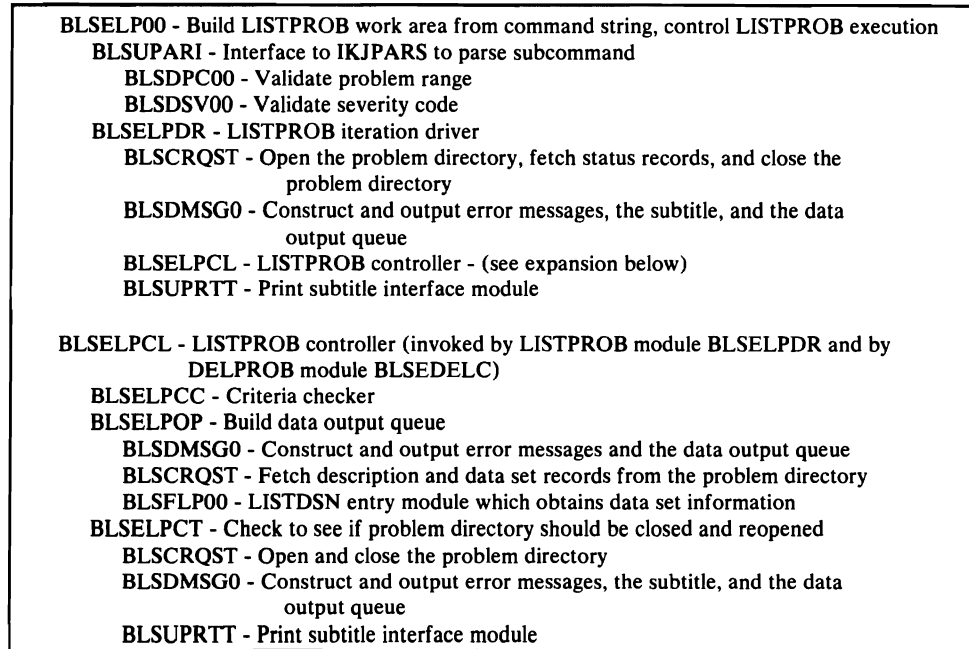


Figure 52. LISTPROB Subcommand Control Flow

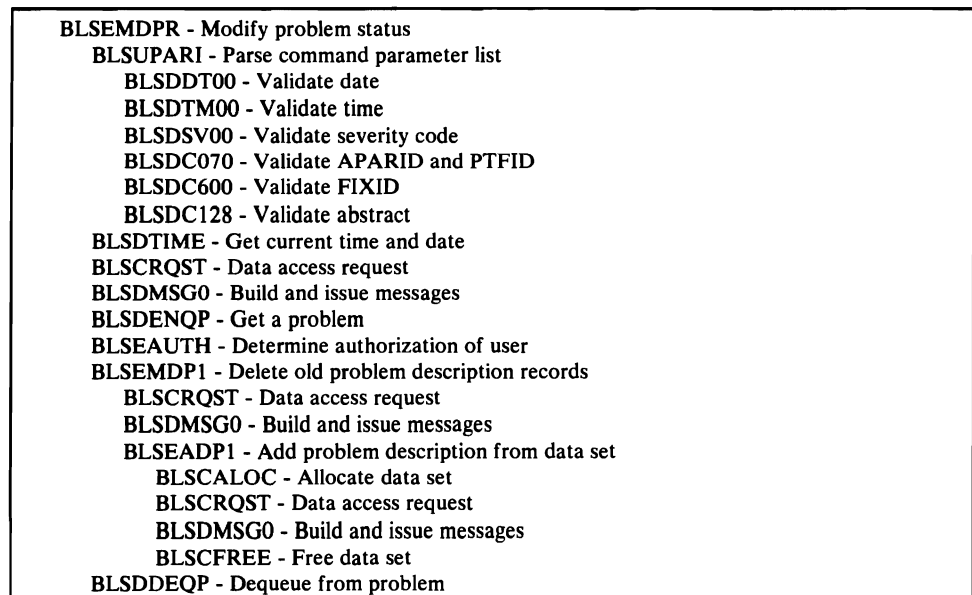


Figure 53. MODPROB Subcommand Control Flow

BLSFAD00 - ADDDSN main processor
 BLSUPARI - Parse subcommand operands
 BLSDC600 - Validate data set description
 BLSFAV00 - Validate the subcommand parameters
 BLSFPS00 - Obtain current problem ID and PDR status record key
 BLSDENQP - Enqueue on problem
 BLSFGP00 - Read PDR status record
 BLSDMSG0 - Issue message: 'problem does not exist'
 BLSCRQST - Data access request
 BLSFDS00 - Get current DSN and member DSD base record key
 BLSDMSG0 - Issue message: 'Forced UNMANAGED attribute'
 BLSFGD00 - Read DSD base record
 BLSFFP00 - Read in DSD problem association and PDR data set association records
 BLSFGD00 - Read record from DSD data base
 BLSFGP00 - Read record from PDR
 BLSFND00 - Build incore DSD base record
 BLSFOD00 - Verify DSD base record
 BLSFDF00 - Format old DSD base record for display
 BLSFCN00 - Request user confirmation
 BLSDMSG0 - Obtain message text
 IKJPTGT - TSO PUT/GET service routine
 BLSFUD00 - Write base record to DSD
 BLSCRQST - Data access request
 BLSFBP00 - Build PDR data set association record
 BLSFND01 - Build DSD problem association record
 BLSFUP00 - Write problem status record to PDR
 BLSCRQST - Data access request
 BLSFUD00 - Write problem association record to DSD
 BLSCRQST - Data access request
 BLSFUP00 - Write data set association records to the PDR
 BLSCRQST - Data access request
 BLSFSD00 - Establish default DSN
 BLSRDUCC - Close and free dump data set
 BLSDMSG0 - Issue message BLS4060I
 BLSDEQ00 - Dequeue from problem

Figure 54. ADDDSN Subcommand Control Flow

BLSFDD00 - DELDSN subcommand processor
 BLSFPARS - Parse the DELDSN subcommand
 BLSUPARI - Parse command
 BLSDMSG0 - Issue a message
 BLSDENQP - Enqueue on the problem
 BLSFVRFY - Verify user's authorization
 BLSCRQST - Get a record
 BLSDMSG0 - Issue a message
 BLSEAUTH - Check user's authorization
 BLSFOPEN - Open a data set
 BLSFLALL - Open PDR, build association list, close PDR
 BLSELPFR - Build list of data sets associated with problem
 BLSCRQST - Get a record, position a data set
 BLSDMSG0 - Issue a message
 BLSFOPEN - Open a data set
 BLSFCLOS - Close a data set
 BLSDMSG0 - Issue a message
 BLSFLONE - Build list of one problem-dataset association
 BLSDMSG0 - Issue a message
 BLSFCONF - Ask user for confirmation of operation
 BLSELPDS - Print data set names on association list
 BLSDMSG0 - Issue a message
 BLSFLP00 - List data set name and information
 BLSDMSG0 - Issue a message
 BLSFGD00 - Get DSD base record
 BLSCRQST - Open data set, close data set, get record
 BLSFFL00 - Print data set name, type, and management
 BLSFSL00 - Print data set description
 BLSFCN00 - Ask user for confirmation
 BLSFDALL - Delete all problem-data set associations on list
 BLSDMSG0 - Issue a message
 BLSFDEL - Delete a problem-data set association from data base
 BLSFOPEN - Open a data set
 BLSCRQST - Get a record
 BLSDMSG0 - Issue a message
 BLSFPRES - Erase a PDR record
 BLSFREAS - Erase a record
 BLSCRQST - Erase a record
 BLSDMSG0 - Issue a message
 BLSDMSG0 - Issue a message
 BLSFREAS - Erase a record
 BLSRDUCC - Close and free dump data set
 BLSFDRES - Erase a record from the DSD
 BLSCRQST - Get a record
 BLSFREAS - Erase a record
 BLSFCLOS - Close a data set
 BLSFSCRT - Scratch a data set
 BLSCALOC - Allocate a data set
 BLSCFREE - Free a data set
 BLSCRQST - Open a data set, close a data set
 BLSDMSG0 - Issue a message
 BLSELPFM - Free storage occupied by association list
 BLSDMSG0 - Issue a message
 BLSDDDEQP - Dequeue the problem

Figure 55. DELDSN Subcommand Control Flow

BLSFLD00 - LISTDSN main processor
 BLSUPARI - Parse command operands
 BLSFDS00 - Setup data set name and key to access DSD base record this data
 set name (called only if data set name keyword specified)
 BLSFGD00 - Get DSD base record for this record (called only if data set name
 keyword specified)
 BLSCRQST - Get DSD base record for this data set name
 BLSDMSG0 - Issue any error messages
 BLSDMSG0 - Output any error messages
 BLSFGG00 - Get first DSN base record in DSD (called only if ALL keyword specified)
 BLSFDF00 - Format and output the DSN attribute listing
 BLSFFL00 - Format data set name, type, and management attribute
 BLSDMSG0 - Build output lines
 BLSFSL00 - Format data set description
 BLSDMSG0 - Build output line
 BLSFTL00 - Format data set problem association list (called only if the
 PROBLEMS keyword is specified)
 BLSFGG00 - Get DSD problem association record
 BLSCRQST - Get DSD problem association record
 BLSDMSG0 - Issue any error messages
 BLSDMSG0 - Build output line
 BLSDMSG0 - Output lines formatted in BLSFFL00, BLSFSL00, and BLSFTL00
 BLSDMSG0 - Output any error messages

Figure 56. LISTDSN Subcommand Control Flow

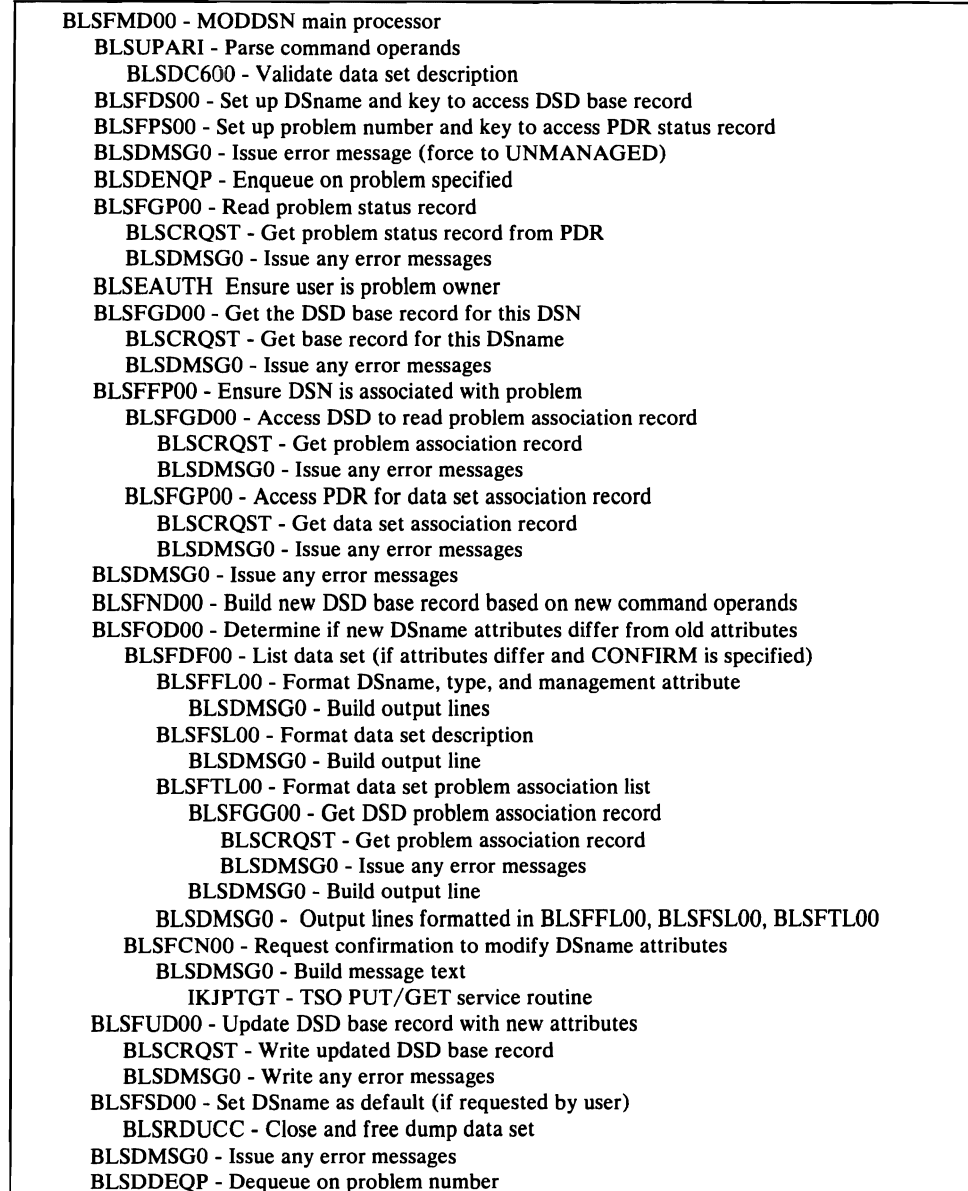


Figure 57. MODDSN Subcommand Control Flow

For lower levels of control flow, refer to the individual module descriptions.

BLSDDDEQP

Function: Problem DEQ

BLSDDDEQP releases control of a problem.

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSDENQP

Function: **Problem ENQ**

BLSDENQP is used to acquire a problem for use, while ensuring that serialization rules for resource acquisition are followed.

Modules Called: BLSDENQ0

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Number of problem to be enqueued.
3. Type of ENQ (1 character):

S SHR
E EXC

Output: None

BLSDENQ0: Enqueue on Resource

Function: BLSDENQ0 enqueues on the problem identifier, or the problem or data set directory.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Major resource name
3. Minor resource name
4. Minor resource name length
5. Literal ‘S’ for share, or ‘E’ for exclusive

Output: Resource enqueued

BLSDTIME: Obtain Current Date and Time

Function: This module obtains the current date (MM/DD/YY) and time (HH:MM:SS) from the host system.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. The TIMECHAR field.
3. The DATECHAR field.

Output

1. The current time.
2. The current date.

BLSEADPR: ADDPROB Subcommand Processor

Function: This module adds a problem to the IPCS data base.

Modules Called: BLSDDT00, BLSDTM00, BLSDSV00, BLSDC600, BLSDC128, BLSDMSG0, BLSDMSG1, BLSDBTIME, BLSEADP1, BLSCRQST, BLSDC070, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPGME, BLSUPGMS, BLSUVAST

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: Updated problem directory data set.

BLSEADP1: ADDPROB Subprocessor

Function: This module is called by BLSEADPR (ADDPROB) or BLSEMDP1 (MODPROB) to copy a description of a problem from a specified data set into the problem directory data set.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Data set name of input data set.
3. Problem status record (see “BLSPDR” on page 357)

Output: Updated problem directory data set.

BLSEAUTH: Problem Management Command Authorization

Function: BLSEAUTH determines whether a user is authorized to use a command based upon the following authorization rules:

Rule	Who is authorized.
Owner	Problem owner or user designated with ADMINAUTHORITY in FPBLOK. The owner rule is intended to be used by the MODPROB, DELDSN, and MODDSN subcommand processors.
Deleter	User designated with delete authority in FPBLOK. If no such designated user, then problem owner or user designated with ADMINAUTHORITY in FPBLOK. The deleter rule is intended to be used by the DELPROB subcommand processor.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Problem status record (see “BLSPDR” on page 357).
3. Name of rule (either “O” or “D”).

Output: None

BLSEDEL: DELPROB Confirmation

Function: BLSEDEL implements the CONFIRM option of the DELPROB subcommand. It assumes that the problem to be deleted has been locked.

Modules Called: BLSELPCL, BLSFCN00

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Problem status record (see “BLSPDR” on page 357).

Output: Confirmation flag (in BLSUZZ2).

BLSEDEL0: DELPROB Subcommand Main Module

Function: BLSEDEL0 is the main module of the delete problem (DELPROB) subcommand processor.

Modules Called: BLSCROST, BLSDDEQP, BLSDENQP, BLSDMSG0, BLSDMSG0, BLSEAUTH, BLSEDEL1, BLSUALLO, BLSUFRE1, BLSUPARI

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: None.

BLSEDEL1: DELPROB Execution Module

Function: BLSEDEL1 deletes problems from the IPCS problem directory. It assumes that the problem to be deleted has been locked.

Modules Called: BLSDMSG0, BLSEDELC, BLSFDDDP, BLSDMSG0, BLSUALLO, BLSUFRE1

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: None.

BLSELPCC: Criteria Checker

Function: This module checks the criteria specified by the user against the information in the current status record.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. LISTPROB work area (LPWA).
3. Problem status record (see “BLSPDR” on page 357)

Output: None

BLSELPCL: LISTPROB Controller

Function: This module calls the criteria checker. If the current problem matches the user's criteria or if there are no selection criteria specified, then it calls module BLSELPPOP to build the data output queue. It calls module BLSELPCT to check if the PDR should be closed and reopened.

Modules Called: BLSELPCC, BLSELPPOP, BLSELPCT, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. LISTPROB work area
3. Problem status record (see “BLSPDR” on page 357)
4. Time at which PDR was opened
5. Number of status records sequentially fetched from the PDR
6. Count of the number of problems meeting the selection criteria
7. Address of the data output queue
8. Number of lines in the data output queue

Output

1. Time at which PDR was opened
2. Number of status records sequentially fetched from the PDR
3. Count of the number of problems meeting the selection criteria
4. Address of the data output queue
5. Number of lines in the data output queue

BLSELPCT: LISTPROB, Check PDR Close/Reopen Criteria

Function: If the number of seconds that the PDR has been open exceeds a constant factor, or the number of lines in the data output queue exceeds a constant factor, or the user specified the default problem, then this module closes the PDR, outputs the data output queue, and reopens the PDR.

Modules Called: BLSCRQST, BLSDMSGs, BLSDMSG0, BLSUALLO, BLSUFRE1, BLSUPRTT

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Time at which PDR was opened.
3. Number of PDR status records sequentially fetched.
4. LISTPROB work area.
5. Address of data output queue.
6. Number of lines in data output queue.
7. Count of problems meeting the selection criteria.

Output: PARMTIME, NUMPROBS, DATAHEAD, and QUEUELNS are updated.

BLSELPDR: Iteration Driver for LISTPROB

Function: This module iteratively fetches a status record and calls BLSELPCL to begin processing the record.

Modules Called: BLSCRQST, BLSDMSGs, BLSDMSG0, BLSELPCL, BLSUALLO, BLSUFRE1, BLSUPRTT

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. LISTPROB work area.

Output: None

BLSELPDS: Output Data Set Attributes

Function: This module constructs and puts out a list of data set names and their attributes based on the data set association list.

Modules Called: BLSDMSG0, BLSFLP00, BLSDMSG0, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Data set association list.
3. Description flag.

Output: None

BLSELPFM: Release Storage Occupied by an Association List

Function: This module releases (via FREEMAIN) the storage obtained for the association list.

Modules Called: BLSDMSG0, BLSDMSG0, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. A 4-byte pointer to the association list whose storage is to be freed.

Output: None

BLSELPFR: Fetch Data Set Associations Records

Function: This module fetches the data set association records for a problem, four at a time, and stores them in an association list (ALELEM) it has created.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. PROPID - eight characters containing the problem identifier whose associations are to be placed in the list.

Output: A pointer to the new association list.

BLSELP0P: LISTPROB Output Module

Function: This module builds the output queue of problem data according to the specified format. Description lines, if required, are fetched from the problem directory. If data set names are required, the list of data set names are retrieved from the problem directory, and module BLSFLP00 is called.

Modules Called: BLSCRQST, BLSMSG0, BLSMSG0, BLSFLP00, BLSUALLO, BLSUFRE1

Input

1. Active task variable(see "BLSUZZ2" on page 406).
2. Problem status record (see "BLSPDR" on page 357)
3. LISTPROB work area.
4. Count of problems meeting the selection criteria.
5. Address of the data output queue.
6. Number of lines in the data output queue.

Output

1. Address of data output queue.
2. Number of lines in the data output queue.

BLSELP00: LISTPROB Main Module

Function: BLSELP00 implements the LISTPROB subcommand. BLSELP00 parses the user-entered subcommand and builds an internal representation of the request (the LPWA). The LPWA consists of header information and a list of selection criterion blocks (SCBs). There is one SCB for each field in the problem status record which is to be used for problem selection. The SCB contains the offset and length of the status record field, and a sorted list of values for the field.

Modules Called: BLSELPDR

Input: Active task variable (see "BLSUZZ2" on page 406).

Output: None

BLSEMDPR: MODPROB Subcommand Processor

Function: This module modifies a problem in the IPCS data base.

Modules Called: BLSDDT00, BLSDDTIME, BLSDDTM00, BLSDSV00, BLSDC070, BLSDC600, BLSDC128, BLSEMDP1, BLSMSG0, BLSMSG0, BLSEAUTH, BLSDENQP, BLSDDDEQP, BLSRZZ3U, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPGME, BLSUPGMS

Input: Active task variable (see "BLSUZZ2" on page 406).

Output: Updated problem directory data set.

BLSEMDP1: MODPROB Subprocessor

Function: This module replaces a problem description in the problem directory with a description from a specified data set.

Modules Called: BLSCRQST, BLSDMSGs, BLSDMSG0, BLSEADP1, BLSUALLO, BLSUFRE1, BLSUPGME

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Name of input data set.
3. Problem status record (see “BLSPDR” on page 357)

Output: Updated problem directory data set.

BLSFAD00: Add Data Set (ADDDSN) Subcommand Processor

Function: This module processes the ADDDSN subcommand. The relationship of the data set to a problem is entered into both the data set and problem directories.

Modules Called: BLSDC600, BLSDDEQP, BLSDMSGs, BLSDMSG0, BLSFAV00, BLSFBP00, BLSFFP00, BLSFGD00, BLSFND00, BLSFND01, BLSFOD00, BLSFSD00, BLSFUD00, BLSFUP00, BLSUALLO, BLSUFRE1, BLSUPARI.

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: Updated data set base record, PDR status record, DSD problem association record, and PDR data set association record.

BLSFAV00: Validate Parameters for Problem Number and Data Set Name

Function: This module establishes if data set name and problem ID are specified in the subcommand. If so, they become the current value. If missing, the default value is obtained from BLSUZZ2.

Modules Called: BLSDENQP, BLSDMSGs, BLSDMSG0, BLSFDS00, BLSFGP00, BLSFPS00, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Data set PDE.
3. Problem PDE.
4. Managed/unmanaged PDE.
5. Area for default data set name.
6. Area for default data set member.
7. Area for default problem ID (8 characters).
8. Area for PDR status record key (see “BLSPDR” on page 357).
9. Area for DSD base record key (see “BLSDSD” on page 342).
10. Area for PDR status record (see “BLSPDR” on page 357).

Output

1. Default data set name or blanks.
2. Default data set member name or blanks.
3. Default problem number (8 characters).
4. PDR status record key.
5. DSD base record key.
6. PDR status record (original copy).

BLSFBP00: Build PDR Data Set Association Record

Function: This module builds the PDR data set association record for the current problem.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Current problem number.
3. Record sequence number.
4. Current data set name.
5. Current data set member name.
6. Area to build PDR data set association record (see “BLSPDR” on page 357).

Output: PDR data set record.

BLSFCLOS: Close a Data Set

Function: This module closes a data set, and, if an error was encountered, it issues appropriate messages and abends, if necessary.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. DMCB of the data set to be closed.
3. Number of the message which is to be issued along with the data access services messages in the event of a failure.

Output: None

BLSFCN00: Confirmation Message and Reply Processor

Function: This module issues a confirmation message to the user using the TSO PUTGET service routine. The reply is checked for a valid answer.

Modules Called: BLSDMSG0, BLSDMSG0, BLSUALL0, BLSUFRE1, IKJPTGT

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Message identifier of text to be put to the terminal.

Output: Answer of Y or N from the user

BLSFCONF: Request Confirmation for DELDSN Subcommand

Function: This module generates a list of data set associations which the DELDSN subcommand would delete. It requests the user to confirm the operation to be performed.

Modules Called: BLSDMSGs, BLSDMSG0, BLSELPDS, BLSFCN00, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Association list whose deletion is to be confirmed.
3. Area for operation indication.

Output: One byte indicating whether the user has confirmed the operation. A value of X'FF' indicates confirmed and X'00' indicates not confirmed.

BLSFDALL: Disassociate All Data Sets on List

Function: This module performs the DELETE function for all associations on a list.

Modules Called: BLSDMSGs, BLSFDEL, BLSDMSG0, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. A list of associations which are to be deleted.
3. Eight characters indicating the password to be used when scratching a data set. If this is all blanks (' '), the user is prompted for the password.

Output: None

BLSFDDDP: Interface from DELPROB to DELDSN

Function: This module performs all of the disassociation and delete operations for the DELPROB subcommand by interfacing to DELDSN modules.

Modules Called: BLSDMSGs, BLSDMSG0, BLSELPFM, BLSUALLO, BLSUFRE1, BLSFLALL, BLSFDALL, BLSFPREA, BLSFOPEN, BLSFCLOS

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Key to the problem status record (see “BLSPDR” on page 357).

Output: None

BLSFDD00: DELDSN Subcommand Processor

Function: This is the top level module for the DELDSN subcommand. It performs the DELDSN subcommand by calling lower level modules.

Modules Called: BLSFPARS, BLSFVRFY, BLSFLALL, BLSFLONE, BLSFCONF, BLSFDALL, BLSELPFM, BLSDENQP, BLSDDEQP, BLSUALLO, BLSUFRE1

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: None

BLSFDEL: Delete a Problem-Data Set Association

Function: This module deletes a problem-data set association from the data base (the PDR and the DSD). If this is the last association with the data set, and if it is the default problem being dissociated, then BLSRDUCC is called to close and free the data set. The data set base record is deleted from the DSD, causing the data set to become unknown to IPCS. Also, if this is the last association and the data set is managed by IPCS, the data set is scratched or initialized.

Modules Called: BLSDMSG0, BLSUALLO, BLSUFRE1, BLSFDRES, BLSFPRES, BLSFSCRT, BLSFOPEN, BLSFCLOS, BLSFREAS, BLSRDUCC, BLSDMSG0

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Eight characters specifying the problem part of the problem-data set association.
3. 52 characters specifying the data set part of the problem-data set association. The first 44 characters contain the data set name and the last eight contain the member name.
4. 8-character password of the data set. All blanks (‘ ’) indicates that the password is unknown, and will be prompted for if needed.

Output: None

BLSFDELS: Delete a Record from the Data Set Directory

Function: This module deletes a specified record from the DSD.

Modules Called: BLSFERAS

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. One character giving the record type.
3. 52 characters identifying the data set. The first 44 characters contain the data set name and the last eight contain the member name.
4. Eight characters identifying the problem. This is ignored if type=‘B’.

5. Four bytes indicating the length of the “record” parameter in bytes. The DSD record to be erased is stored left-justified, truncated or padded-right with blanks if necessary, in the first RECLen bytes of the “record” parameter. If $\text{RECLen} \leq 0$, then RECORD remains unchanged.

Output

1. A storage area in which a copy of the erased record is to be returned. This parameter is ignored if $\text{RECLen} \leq 0$. See the “RECLen” parameter under input.
2. A 4-byte pointer to the message chain built by this routine. The chain contains messages if return code is 4, otherwise MSGPTR=0.

BLSFDF00: LISTDSN Attributes Driver

Function: This module processes the listing of DSN attributes for IPCS data management. The type, management attribute, and data set name are formatted. Next, the DSN description is formatted, and optionally, additional lines are formatted containing the list of problems with which the DSN is associated.

Modules Called: BLSFFL00, BLSFSL00, BLSFTL00, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. The DSD base record for the DSN to be formatted.
3. Three characters containing YES or NO to control the problem list formatting.
4. Pointer to the report queue containing the report lines.

Output: An attribute listing appended to the end of the input report line queue.

BLSFDS00: Build VSAM Key for DSD Base Record

Function: This module establishes the current data set name and member, and builds the VSAM key for DSD base record using these values.

Modules Called: BLSDMSG0, BLSDMSG0, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Data set name PDE.

Output

1. Key for DSD base record (see “BLSDSD” on page 342).
2. Current data set name (44 characters).
3. Current data set member name (8 characters).

BLSFREAS: Erase a Record from a Data Set

Function: This module erases a record from a data set, and if an error was encountered, issues a message and abends, if necessary. A GET for update has already been done for the record which is to be erased.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. DMCB (see “BLSDMCB” on page 336). to be erased.
3. Number of the message which is to be issued along with the data access services messages in the event of a failure.

Output: None

BLSFFL00: LISTDSN First Formatting Routine

Function: This module lists the DSN attributes of type (managed or unmanaged) and the data set name on the first line of the display.

Modules Called: BLSDMSGs, BLSDMSG0, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. DSD base record (see “BLSDSD” on page 342).
3. Pointer to report line queue.

Output: First lines of the report for LISTDSN are appended to the end of the report line queue.

BLSFFP00: Obtain PDR Data Set Record from IPCS Data Base

Function: This module provides the DSD problem and the PDR data set association records when called.

Modules Called: BLSFGP00, BLSFGD00, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Current data set name.
3. Current data set member.
4. Current problem ID.

Output

1. DSD problem association record (see “BLSDSD” on page 342).
2. PDR data set record (see “BLSPDR” on page 357).

BLSFGD00: Get DSD Record

Function: This module gets a DSD record (based on the key passed) from the IPCS data base. Expected conditions are “record found” and “no record found.”

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Key of DSD record (see “BLSDSD” on page 342).
3. An area to return the record retrieved (see “BLSDSD” on page 342).

Output: None

BLSFGG00: Get DSD Record Key

Function: This module gets a DSD record from the IPCS data base whose key is greater than or equal to the key passed.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Key of DSD record (see “BLSDSD” on page 342).
3. An area to return the record retrieved (see “BLSDSD” on page 342).

Output: None

BLSFGP00: Get PDR Record

Function: This module gets a PDR record from the IPCS data base. Expected conditions are “record found” and “no record found.”

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Key of PDR record (see “BLSPDR” on page 357).
3. An area to return the record retrieved (see “BLSPDR” on page 357).

Output: None

BLSFLALL: Build an Association List of Data Sets

Function: This module builds a list containing all data sets associated with a problem. A message is issued if none exist.

Modules Called: BLSDMSG0, BLSELPFR, BLSFOPEN, BLSFCLOS, BLSDMSG0, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. 8-character name of the problem whose data set associations are to be put on the list.

Output: A 4-byte pointer to the association list produced by this procedure. A zero indicates the empty list.

BLSFLD00: LISTDSN Subcommand Processor

Function: This module sends the DSN attributes to the user terminal, the print data set, or both.

Modules Called: BLSDMSG0, BLSFGD00, BLSFGG00, BLSFDF00, BLSDMSG0, BLSFDS00, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPARU, BLSUPUTA

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSFLONE: Build a List Containing One Association

Function: This module builds an association list containing one element as specified by the caller.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Eight characters containing the problem to be placed in the association.
3. 52 characters containing the data set name to be placed in the association. The first 44 characters contain the data set name; the last eight contain the member name.

Output: 4-byte pointer to the association list which has been built. If no list has been built (return code 4), this is zero.

BLSFLP00: List Data Set Attributes

Function: This module processes the listing of DSN attributes for the LISTPROB subcommand. The type, management attribute, and data set name are formatted on the first line. An optional second line containing the DSN description is also formatted.

Modules Called: BLSDMSG0, BLSFGD00, BLSDMSG0, BLSFFL00, BLSFSL00, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. The DS name to be formatted.
3. Three characters containing YES or NO indicating whether the 60-character data set name description should be included in the listing.

Output: None

BLSFMD00: MODDSN Subcommand Processor

Function: This module processes the MODDSN subcommand. The attributes of the DSN are modified and noted in the data set directory.

Modules Called: BLSDC600, BLSDDEQP, BLSDENQP, BLSDMSG0, BLSDMSG0, BLSEAUTH, BLSFDS00, BLSFFP00, BLSFGD00, BLSFGP00, BLSFND00, BLSFOD00, BLSFPS00, BLSFSD00, BLSFUD00, BLSUALLO, BLSUFRE1, BLSUPARI

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: An updated data set directory.

BLSFND00: Build DSD Base Record

Function: This module builds the DSD base record for a data set that is being entered into the IPCS data base for the first time.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Key for the DSD base record (see “BLSDSD” on page 342)
3. Type field
4. MANAGED or UNMANAGED parameter
5. Description

Output: DSD base record

BLSFND01: Build DSD Data Set Association Record

Function: This module builds the DSD problem association record.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Current data set name
3. Current data set member name
4. Current problem number
5. PDR sequence number

Output: Data set association record (see “BLSDSD” on page 342)

BLSFOD00: Validate DSD Base Record for Conflicting Attributes

Function: This module verifies that all attributes specified in the ADDDSN subcommand do not conflict with attributes previously specified when the data set was first added to the data base.

Modules Called: BLSDMSG0, BLSDMSG0, BLSFCN00, BLSFDF00, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Original DSD base record (see “BLSDSD” on page 342)
3. New DSD base record (see “BLSDSD” on page 342)
4. Parse descriptor for DSN attribute
5. Parse descriptor for type subfield
6. Parse descriptor for MANAGED/UNMANAGED attribute
7. Parse descriptor for description subfield

Output: Verified DSD base record

BLSFOPEN: Open a Data Set

Function: This module opens a data set, and if an error was encountered issues appropriate messages, and abends, if necessary.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. A 4-byte pointer to the DMCB of the data set to be closed.
3. Four bytes containing the number of the message which is to be issued along with the data access services messages in the event of a failure.
4. A one-byte flag indicating if the data set is to be opened for output. If output is X'FF', then the data set is opened for output, otherwise it is opened for input.

Output: None

BLSFPARS: Parse the DELDSN Subcommand

Function: This module parses the keywords and keyword values in the command buffer and returns them in the parameter list. Default parameters are supplied where appropriate.

Input: Active task variable (see “BLSUZZ2” on page 406)

Output

1. Eight characters containing the problem ID.
2. 52 characters identifying the data set. The first 44 characters contain the data set name and the last eight contain the member name. If ALL was specified, this parameter's value is undefined.
3. Eight characters containing password or data set. If a default data set is used or if the password is not specified, the value returned is all blanks. Its value is undefined if ALL is specified.
4. A one-byte binary value indicating if ALL was specified
X'00' ALL specified.
X'FF' ALL not specified.
5. A one-byte binary value indicating if CONFIRM was specified
X'00' CONFIRM specified.
X'FF' CONFIRM not specified.
6. A one-byte binary value indicating if TEST was specified
X'00' TEST specified.
X'FF' TEST not specified.

BLSFPERA: Delete a Sequence of Records from the PDR

Function: This module deletes all records with a specified type and problem regardless of sequence number from the PDR.

Modules Called: BLSFERAS

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. 2-character record type.
3. 8-character problem identifier.

Output: A 4-byte pointer to the message chain built by this routine. The chain contains messages if return code is 4, otherwise MSGPTR = 0.

BLSFPERS: Delete a Record from the PDR

Function: This module deletes a specified record from the PDR.

Modules Called: BLSFERAS

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. 2-character record type.
3. 8-character problem id.
4. 4-character sequence number of the record.

Output: A 4-byte pointer to the message chain built by this routine. The chain contains messages if return code is 4. Otherwise, MSGPTR = 0.

BLSFPS00: Build PDR Status Record Key

Function: This module establishes the current problem number and uses it to build the VSAM key for accessing the PDR status record.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Problem number PDE.
3. An area for the current problem ID (8 characters).
4. An area to build the key. (see “BLSPDR” on page 357)

Output

1. VSAM key for PDR status record for current problem
2. Current problem number

BLSFSCRT: Scratch Data Set

Function: This module scratches or initializes a managed data set.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. 52-character DS name to be deleted
 - a. 44-character DS name
 - b. 8-character member name
3. 8-character WRITE password or blanks

Output: None

BLSFSD00: Establish Data Set and Member Names as Defaults

Function: This module makes the current data set name, member name, and password the default values, and records that default values have changed (via return code to BLSRDUCC).

Modules Called: BLSDMSGs, BLSDMSG0, BLSRZZ3U, BLSUALLO, BLSUDDSN, BLSUFRE1, BLSUPGMS

Input

1. Parse descriptor entry for data set name.
2. Active task variable (see “BLSUZZ2” on page 406).

Output

1. Updated default data set name
2. Updated default member name
3. Indication that defaults have been updated (RC=0)

BLSFSL00: LISTDSN Second Formatting Routine

Function: This module lists the description for the data set name.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. DSD base record (see “BLSDSD” on page 342)
3. Pointer to the report line queue.

Output: Description part of the LISTDSN report appended to the end of the report line queue.

BLSFTL00: LISTDSN Third Formatting Routine

Function: This module lists the problems associated with a data set.

Modules Called: BLSFSG00

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. DSD base record (see “BLSDSD” on page 342)
3. Pointer to report line queue.

Output: One or more lines of problem association information appended to the end of the report line queue.

BLSFUD00: Write a Record to the DSD Data Base

Function: This module writes a record to the DSD data base and closes the file after having previously opened it.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. DSD record to be written (see “BLSDSD” on page 342)
3. DSD record length

Output: DSD record written to DSD data base

BLSFUP00: Write a Record to PDR Data Set

Function: This module opens the PDR data set for update, ensures data set integrity, writes the record, and closes the PDR data set.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. PDR record to be written (see “BLSPDR” on page 357)
3. PDR record length

Output: PDR record written to PDR data base

BLSFVRFY: Verify Validity of DELDSN Operation

Function: This module checks the user’s authority to delete data. If the user is not authorized, a message is issued. If the authority cannot be checked because of a missing problem status record, the module considers the user authorized and issues a message.

Modules Called: BLSEAUTH, BLSFOPEN, BLSFCLOS

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. 8-character problem identifier whose association is to be deleted.

Output: One byte. If the return code is 0, then X’00’ indicates that authority has not been verified and X’FF&eeq1. indicates that it has been verified. If the return code is not 0, then this value is undefined.

Analysis Subcommands

The analysis subcommands perform checks and display information from the various components specified. The program control flow charts for the five subcommand processors are shown in Figure 58 through Figure 62 on page 198.

ASMCHECK Subcommand

The program control flow showing the modules involved in processing the ASMCHECK subcommand is presented in Figure 58 .

BLSRASM7 - IPCS ASMCHECK main processor
BLSUALLO - Allocate automatic storage
BLSUPARI - IKJPARS interface
BLSUPARU - Process routing operands
BLSRDUSO - Prepare dump for use
BLSRESGU - Retrieve active equate symbol record
BLSRESAR - Add or replace equate symbol record
BLSUMPK1 - Compress a message
BLSUPUTO - Route single-level message
BLSRACC - Debugging storage access service routine
BLSUFRE1 - Free automatic storage

Figure 58. ASMCHECK Subcommand Control Flow

For lower levels of control flow (for example, calls to common service routines), refer to the individual module descriptions.

BLSRASM7: ASMCHECK Subcommand Processor

Function: BLSRASM7 performs the following processing: obtains and displays the ASMT address via the CVT; receives, processes and displays I/O requests; obtains the page address resolution table (PART) and PART entries; issues messages indicating where page data sets reside; displays any active I/O information; and validity checks some control blocks.

Modules Called: BLSRACC, BLSRDUSO, BLSRESAR, BLSRESGU, BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUPARI, BLSUPARU, BLSUPUTO, BLSUPUTV

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

Data Areas Used: ASMT, CVT, IORB, PART, IOSB, UCB

COMCHECK Subcommand

The program control flow for processing the COMCHECK subcommand is shown in Figure 59 on page 193.

BLSRCOMK - IPCS COMCHECK main processor
BLSUALLO - Allocate automatic storage
BLSUPARI - IKJPARS interface
BLSUPARU - Process routing operands
BLSRDUSO - Prepare dump for use
BLSRM077 - Transmit message BLS18077I to warn about virtual dump processing
BLSRCOML - Count WQEs
BLSUALLO - Allocate automatic storage
BLSRESGU - Retrieve active equate symbol record
BLSRACC - Debugging storage access service routine
BLSUMPK1 - Compress a message
BLSUPUTO - Route single-level message
BLSUFRE1 - Free automatic storage
BLSUFRE1 - Free automatic storage

Figure 59. COMCHECK Subcommand Control Flow

For lower levels of control flow, refer to the individual module descriptions.

BLSRCOMK: COMCHECK Subcommand Processor

Function: BLSRCOMK performs the following processing: counts the write-to-operator queue elements (WQEs) chained from the unit control module (UCM) and compares the total to the UCMWQNR field; checks the status byte for each console within each unit control module element (UCME); obtains the unit control block (UCB) by issuing one or more messages; and writes any outstanding WTOR numbers and text.

Modules Called: BLSRACC, BLSRCOML, BLSRESGU, BLSRDUSO, BLSRM077, BLSUALLO, BLSUFRE1, BLSVMPK1, BLSUPARI, BLSUPARU, BLSUPUTO

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

Data Areas Used: UCB, UCM, WQE, CTM(CQES), ORE

BLSRCOML: Comcheck Processor

Function: Count WQE's chained from the UCM, compare the total with UCMWQNR. Follow the UCME's, check the status byte for each console, obtain the UCB, and transmit messages to indicate results. Write any outstanding WTOR numbers or text.

Modules Called: BLSRACC, BLSRACCN, BLSRESSA, BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUPUTV

Input: The active task variable (see “BLSUZZ2” on page 406)

Output: None

Data Areas Used: UCB, UCM, WQE, CQE, ORE

ENQCHECK Subcommand

The program control flow showing the modules involved in processing the ENQCHECK subcommand is shown in Figure 60 .

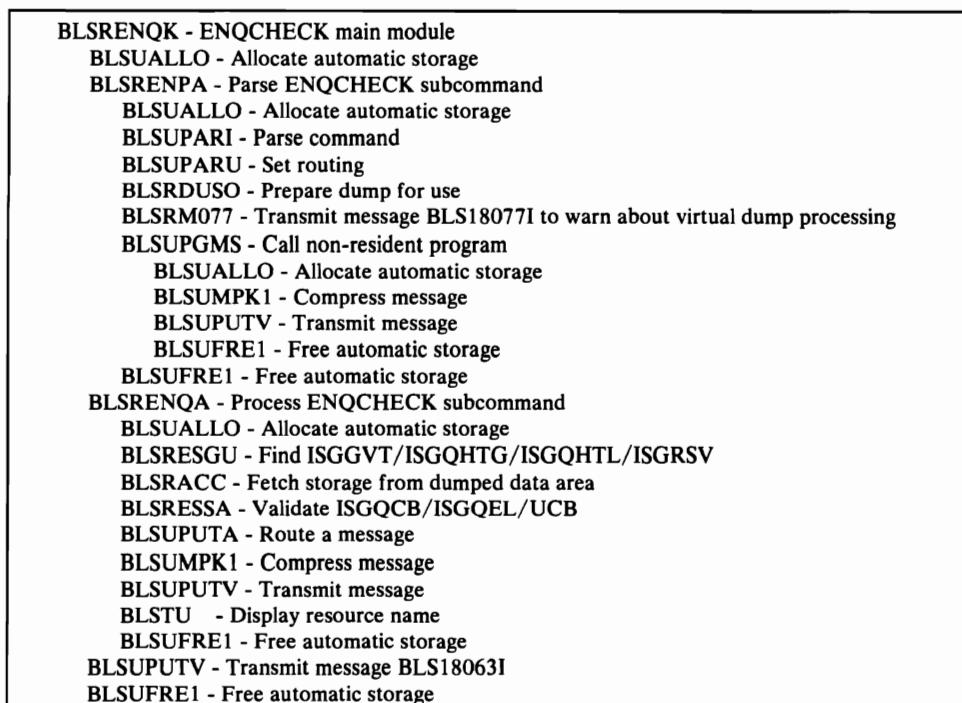


Figure 60. ENQCHECK Subcommand Control Flow

For lower levels of control flow, refer to the individual module descriptions.

BLSRENQK: ENQCHECK Subcommand Processor

Function: BLSRENQK passes control to BLSRENPA to parse the ENQCHECK subcommand and then passes control to BLSRENQA to process the subcommand.

Modules Called: BLSRENPA, BLSRENQA, BLSUALLO, BLSUFRE1, BLSUPUTV

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSRENPA: Parse ENQCHECK Subcommand

Function: BLSRENPA parses the ENQCHECK subcommand and passes control to BLSRM077 to transmit message BLS18077I.

Modules Called: BLSRDUSO, BLSRM077, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPARU, BLSUPGMS

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. ENQCHECK work area

Output: ENQCHECK work area indicates the subcommand options

BLSRENQA: Process ENQCHECK Subcommand

Function: This module examines the ENQ/DEQ control blocks in the dump and produces the requested report regarding ENQ/DEQ resources.

Modules Called: BLSRACC, BLSRESGU, BLSRESSA, BLSTU, BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUPUTA, BLSUPUTV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. ENQCHECK work area

Output: None

Data Areas Used: ISGQCB, ISGQEL, ISGQHT, ISGRSL, ISGRSV, UCB

BLSRM077: Transmit BLS18077I

Function: This module determines whether a virtual dump is being processed. If one is, message BLS18077I is transmitted.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUPUTV

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

STATUS Subcommand

The program control flow showing the modules involved in processing the STATUS subcommand is presented in Figure 61 on page 196 .

BLSRST00 - IPCS STATUS main processor
BLSUALLO - Allocate automatic storage
BLSUPARI - IKJPARS interface
BLSRVPCP - Parse validity check exit for CPU address
BLSUPARU - Process routing operands
BLSRDUSO - Prepare dump for use
BLSRST01 - Process routing operands
BLSUALLO - Allocate automatic storage
BLSUPUTA - Route a message
BLSRESGU - Retrieve definition of symbol
BLSRACC - Get dump data
BLSUPUTV - Transmit message
BLSUMTOD - Time of day processor
BLSUMPK1 - Compress message
BLSRESSA - Scan block
BLSRSSSC - Compress symbol
BLSRZALC - Translate binary to mod26
BLSUPGMS - Call non-resident program
BLSRSUMM - Summary subcommand processor
BLSTR - Display registers
BLSUFRE1 - Free automatic storage
BLSUFRE1 - Free automatic storage

Figure 61. STATUS Subcommand Control Flow

For lower levels of control flow (for example, calls to common service routines), refer to the individual module descriptions.

BLSRST00: STATUS Subcommand Processor

Function: BLSRST00 performs a basic system analysis routine by displaying the nucleus member name, time-of-day clock values (both hardware and software), PSW, general purpose registers, control registers, current ASID, ASCB address, TCB address, and jobname.

Modules Called: BLSRDUSO, BLSRST01, BLSRVPCP, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPARU

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSRST01: Status Processor

Function: Display critical information pertaining to system status, CPU status, or both.

Modules Called: BLSRACC, BLSRESGU, BLSRESSA, BLSRSSSC, BLSRSUMM, BLSRZALC, BLSRZALN, BLSTR, BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUMTOD, BLSUPGMS, BLSUPUTA, BLSUPUTV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. STATUS subcommand work area

Output: None

BLSRZALN: Translate Base-26 To Binary

Function: Translate a base-26 EBCDIC suffix to a binary integer.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Suffix to be translated
3. Unsigned binary fullword

Output: The unsigned binary fullword contains the number described by the suffix.

SUMMARY Subcommand

The program control flow showing the modules involved in processing the SUMMARY subcommand is presented in Figure 62 .

Note: In the figure, when BLSQROUT is called, the requested common exit service code is indicated in parentheses.

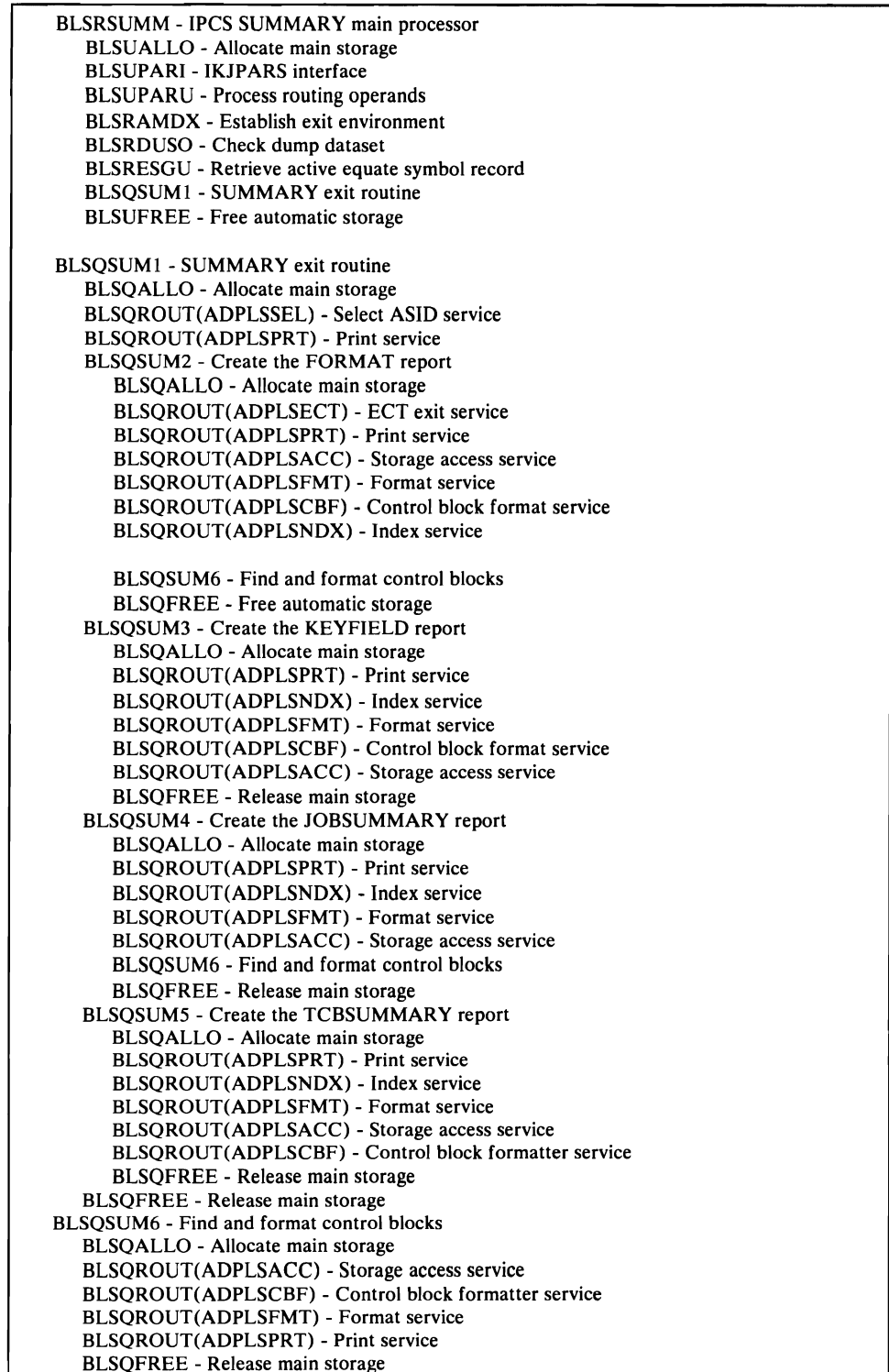


Figure 62. SUMMARY Subcommand Control Flow

For lower levels of control flow refer to the individual module descriptions.

BLSRSUMM: SUMMARY Subcommand Processor

Function: BLSRSUMM performs the following processing: parses operands that require the routing and controlling of output and any keywords by linking to IKJPARS; fills in the internal summary parameter list (BLSQSMPL) with the PARSE results; establishes the exit environment by calling BLSRAMDX; and calls BLSQSUM1 via a link to BLSRAMDX.

Modules Called: BLSQSUM1, BLSRAMDX, BLSRDUSO, BLSRESGU, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPARU

Services Invoked: None

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Input character string

Output: The BLSQSMPL parameter list initialized.

Data Areas Used: None

BLSQSUM1: SUMMARY Subcommand Processor Front End

Function: BLSQSUM1, which is an exit program, is called to format control blocks in the dump according to the format control and selection keywords specified in the input character string. BLSQSUM1 calls the select ASID service to get a list of ASIDs to process and to place the address of the list in BLSQSMPL. When either JOBSUMMARY or FORMAT is specified, BLSQSUM1 obtains the CVT from the dump and extracts and stores several pointers in the BLSQSMPL. Then the appropriate report processing programs are called, passing them the BLSABDPL and BLSQSMPL.

Modules Called: BLSQALLO, BLSQFREE, BLSQROUT, BLSQSUM2, BLSQSUM3, BLSQSUM4, BLSQSUM5

Services Invoked: Print and select ASID

Input

1. BLSABDPL
2. BLSQSMPL

Output: None

Data Areas Used: None

BLSQSUM2: FORMAT Report Generator

Function: BLSRSUM1 calls BLSQSUM2, which is an exit program, to create the FORMAT report. This report contains the major control blocks associated with the ASCBs in the ASID list.

Modules Called: BLSQALLO, BLSQFREE, BLSQROUT, BLSQSUM6

Services Invoked: ECT, control block formatter, format, index, print and storage access

Input

1. BLSABDPL
2. BLSQSMPL

Output: FORMAT report

Data Areas Used: None

BLSQSUM3: KEYFIELD Report Generator

Function: BLSRSUM1 calls BLSQSUM3, which is an exit program, to create the KEYFIELD report. The report contains the key fields from the ASCBs, TCBs, PRBs, and CDEs associated with the ASCBs in the ASID list.

Modules Called: BLSQALLO, BLSQFREE, BLSQROUT

Services Invoked: Control block formatter, format, index, print and storage access index

Input

1. BLSABDPL
2. BLSQSMPL

Output: KEYFIELD report

Data Areas Used: None

BLSQSUM4: JOBSUMMARY Report Generator

Function: BLSRSUM1 calls BLSQSUM4, which is an exit program, to create the JOBSUMMARY report. The report contains a summary of system information, a summary of the ASCBs and TCBs associated with the ASCBs in the ASID list, and a problem list.

Modules Called: BLSQALLO, BLSQFREE, BLSQROUT, BLSQSUM6

Services Invoked: Format, index, print and storage access

Input

1. BLSABDPL
2. BLSQSMPL

Output: JOBSUMMARY report

Data Areas Used: None

BLSQSUM5: TCBSUMMARY Report Generator

Function: BLSRSUM1 calls BLSQSUM5, which is an exit program, to create the TCBSUMMARY report. The report contains a summary of the ASCBs and TCBs associated with the ASCBs in the ASID list.

Modules Called: BLSQALLO, BLSQFREE, BLSQROUT

Services Invoked: Control block formatter, format, index, print and storage access

Input

1. BLSABDPL
2. BLSQSMPL

Output: TCBSUMMARY report

Data Areas Used: None

BLSQSUM6: Find and Format Routine

Function: BLSRSUM2 and BLSQSUM4 call BLSQSUM6, which is an exit program, to process action lists specifying how the control blocks are connected and formatting options.

Modules Called: BLSQALLO, BLSQFREE, BLSQROUT

Services Invoked: Control block formatter, format, print and storage access format

Input

1. BLSABDPL
2. BLSQSMPL

Output: Formatted control blocks and messages

Data Areas Used: None

Exit Interface Services

The common services pass control to a PRDMP exit routine. The following modules provide the interface to invoke PRDMP's exit services.

BLSRAMDX: Invoke PRDMP's Exit

Function: Passes control to a PRDMP exit routine.

Modules Called: AMDPRECT, BLSQROUT, BLSUALLO, BLSUFRE1, BLSUTRMV

Input

1. Active task variable (see "BLSUZZ2" on page 406)
2. An IKJIDENT PDE for the exit program name
3. An IKJIDENT INTEGER PDE for the address mask

Output: None

BLSRPRXC: Create PRDMP Exit Data

Function: Prepare the PRDMP exit simulation data in the active task variable (see "BLSUZZ2" on page 406) and the PRDMP exit data (PRX).

Modules Called: BLSQEXTI, BLSRACCL, BLSRESGU, BLSUALLO, BLSUFRE1

Input: Active task variable (see "BLSUZZ2" on page 406)

Output: None

Exit Support Subcommands

The exit support subcommands pass control to user-exit programs designed to be run under PRDMP. The program control flow for exit subcommands is shown in Figure 63 through Figure 66 on page 204.

ASCBEXIT Subcommand

The program control flow showing the modules involved in processing the ASCBEXIT subcommand is presented in Figure 63 .

BLSRASCX - IPCS ASCBEXIT main processor
BLSUALLO - Allocate automatic storage
BLSUPARI - IKJPARS interface
BLSRVPAS - Parse validity check exit for ASID
BLSUPARU - Set routing
BLSRDUSO - Prepare dump for use
BLSUPGMR - Call non-resident program
BLSRPRXC - Create AMDPRDMP exit data
BLSUPGME - Call non-resident program
BLSRAMDX - Invoke PRDMP Exit module - Specified PRDMP ASCB exit routine
BLSUFRE1 - Free automatic storage

Figure 63. ASCBEXIT Subcommand Control Flow

For lower levels of control flow, refer to the individual module descriptions.

BLSRASCX: ASCBEXIT Subcommand Processor

Function: BLSRASCX executes an ASCB exit routine. BLSRASCX parses the command, fills in the exit parameter list (part of the BLSUZZ2 control block), and passes control to a specified exit routine.

Modules Called: BLSRAMDX, BLSRDUSO, BLSRPRXC, BLSRVPAS, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPARU, BLSUPGME, BLSUPGMR

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

IOSCHECK Subcommand

The program control flow chart for the IOSCHECK subcommand is shown in Figure 64.

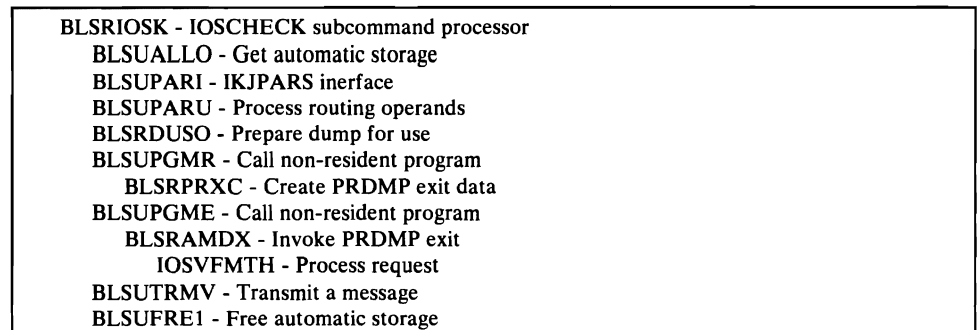


Figure 64. IOSCHECK Subcommand Control Flow

For lower levels of control flow (for example calls to common service routines), refer to the individual module descriptions.

BLSRIOSK: IOSCHECK Subcommand Processor

Function: BLSRIOSK summarizes the system I/O activity described in a dump.

Modules Called: BLSRAMDX, BLSRDUSO, BLSRPRXC, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPARU, BLSUPGME, BLSUPGMR, BLSUTRMV, IOSVFMTH

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

TCBEXIT Subcommand

The program control flow showing the modules involved in processing the TCBEXIT subcommand is presented in Figure 65 on page 204 .

BLSRTCBX - IPCS TCBEXIT main processor
 BLSUALLO - Allocate automatic storage
 BLSUPARI - IKJPARS interface
 BLSRVPAS - Parse validity check exit for ASID ident PDE
 BLSRVPCP - Parse validity check exit for CPU ident
 BLSRVPAD - Address validity check subroutine
 BLSUVP31 - Validate integers 1 through 2^{31}
 BLSUVP32 - Validate integers 0 through $2^{31}-1$
 BLSUVP33 - Validate integer 0 through $2^{24}-1$
 BLSUPARU - Set routing
 BLSRADDs - Resolve address PDEs
 BLSUPGMR - Call non-resident program
 BLSRPRXC - Create PRDMP exit data
 BLSUPGME - Call non-resident program
 BLSRAMDX - Invoke PRDMP Exit
 BLSUFRE1 - Free automatic storage

Figure 65. TCBEXIT Subcommand Control Flow

For lower levels of control flow (for example, calls to common service routines), refer to the individual module descriptions.

BLSRTCBX: TCBEXIT Subcommand Processor

Function: BLSRTCBX executes a PRDMP exit routine. BLSRTCBX parses the command, fills in the exit parameter list (in control block BLSUZZ2), and invokes a specified TCB exit routine.

Modules Called: BLSRADDs, BLSRAMDX, BLSRPRXC, BLSRVPAD, BLSRVPAS, BLSRVPCP, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPARU, BLSUPGME, BLSUPGMR, BLSUVP31, BLSUVP32, BLSUVP33

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

VERBEXIT Subcommand

The program control flow showing the modules involved in processing the VERBEXIT subcommand is presented in Figure 66 .

BLSRVRBX - IPCS VERBEXIT main processor
 BLSUALLO - Allocate automatic storage
 BLSUPARI - IKJPARS interface
 BLSUPARU - Set routing
 BLSRDUSO - Prepare the dump
 BLSUPGMR - Call non-resident program
 BLSRPRXC - Create PRDMP exit data
 BLSUPGME - Call non-resident program
 BLSRAMDX - Invoke PRDMP exit
 Module - Specified PRDMP user control statement exit routine
 BLSUTRMV - Issue message to terminal
 BLSUFRE1 - Free automatic storage

Figure 66. VERBEXIT Subcommand Control Flow

For lower levels of control flow (for example, calls to common service routines), refer to the individual module descriptions.

BLSRVRBX: VERBEXIT Subcommand Processor

Function: BLSRVRBX executes a user-written control statement exit routine. BLSRVRBX parses the command, fills in the exit parameter list (in control block BLSUZZ2), and invokes the specified exit.

Modules Called: BLSRAMDX, BLSRDUSO, BLSRPRXC, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPARU, BLSUPGME, BLSUPGMR, BLSUTRMV

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

Common Exit Services

IPCS, PRDMP, and SNAP access the common exit services (control block formatter, equate symbol, exit control table (ECT), format, get symbol, index, print, select ASID, and storage access) via module BLSQROUT. The following table (Figure 67) shows the service code constants that are in the exit parameter list (BLSABDPL), the host (IPCS, PRDMP, SNAP) that supports the service, and the entry points that process the requested service.

Request Code	SNAP MVS/XA	PRDMP MVS/XA	PRDMP MVS/370	IPCS MVS/XA	IPCS MVS/370
ADPLSACC	Note 1	AMDPRGSA	AMDMEMAR	BLSRACCI	BLSRACCL
ADPLSPRT	IEAVADG3	AMDPRGPR	AMDWRITR	BLSUPUTI	BLSUPUTT
ADPLSFMT	BLSQIFMT	BLSQIFMT	BLSQIFMT	BLSQIFMT	BLSQIFMT
ADPLSCBF	BLSQCFMT	BLSQCFMT	BLSQCFMT	BLSQCFMT	BLSQCFMT
ADPLSNDX	Note 1	AMDPRGIN	AMDINDEX	Note 2	Note 2
ADPLSECT	N/A	BLSQECT	BLSQECT	BLSQECT	BLSQECT
ADPLSGTS	N/A	N/A	N/A	BLSQESGU	BLSQESGU
ADPLSEQU	N/A	N/A	N/A	BLSQESAR	BLSQESAR
ADPLSSEL	N/A	BLSQSEL	BLSQSEL	BLSQSEL	BLSQSEL

Figure 67. Common Exit Services Control Flow

Notes:

1. SNAP provides a NOP condition via entry points contained in modules IEAVADIN, IEAVAD0C, IEAVAD03, and IEAVAD08.
2. IPCS provides a NOP condition via an entry point contained in module BLSRPRXC.

For lower levels of control flow refer to the individual module descriptions.

BLSQROUT: Exit Services Router

Function: BLSQROUT invokes a service indicated by a passed constant in the BLSABDPL.

Modules Called: BLSQALLO, BLSQCFMT, BLSQFREE, BLSQECT, BLSQIFMT, BLSQSEL, BLSQESAR, BLSQESGU

Services Invoked: Control block formatter, ECT, equate symbol, format, get symbol, print, select ASID, and storage access

Input

1. BLSABDPL - exit parameter list
2. SCODE - requested service code
3. SPARM - requested service parameter list

Output: SRACODE - set to 4 if the service is not supported by the host.

BLSQEXTI: Load Exit Services

Function: BLSQEXTI loads the exit service modules, initializes the part of the exit interface that is common to IPCS, PRDMP, and SNAP with the addresses of the services, and deletes the exit service modules.

Modules Called: AMDPRUIM

Services Invoked: None

Input: Fields in the BLSABDPL - exit parameter list

1. ADPLSBPL - subpool for getmain
2. ADPLPRNT - address of print service

Fields in the EXTP parameter list

1. EXTPFUNC - function requested
2. EXTPSERV - services to be initialized
3. EXTPRETN - return area for load list

Output

1. BLSABDPL and BLSQSRA initialized with the addresses of the common exit services and the default formatting columns
2. EXTPRETN - address of the load list containing the addresses and lengths of the common exit services

BLSQCBAT: Control Block Acronym Table

Function: BLSQCBAT is the control block acronym table (CBAT), which associates system acronyms with corresponding models and formatting modules. The CBAT is searched by BLSQCFMT when a request is made to format a system control block.

Modules Called: None

Services Invoked: None

Input: Not applicable

Output: Not applicable

BLSQALLO: Allocate Automatic Storage

Function: BLSQALLO allocates automatic storage for the common exit services modules.

Modules Called: Modules to perform the requested common exit service

Services Invoked: None

Input

Register 9 - contains the address of BLSABDPL

Register 10 - contains the address of BLSQSRA

Output: The address of the storage allocated is returned in register 1.

BLSQFREE: Release Automatic Storage

Function: BLSQFREE releases automatic storage that was obtained for processing of the common exit services modules.

Modules Called: None

Services Invoked: None

Input: Register 1 - contains the address of BLSABDPL

Output: None

BLSQSIGR: Signed Integer Range

Function: BLSQSIGR translates data from EBCDIC to a signed binary integer range that is described by an IKJIDENT PDE generated by IKJPARS.

Modules Called: BLSUALLO, BLSUFREE

Services Invoked: None

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. An IKJIDENT PDE generated by IKJPARS
3. Two fullwords that will contain the translated data requested by the caller

Output

1. A fullword signed binary integer defines the origin of the range
2. A fullword signed binary integer defines the end of the range

BLSQVPAS Validate MVS ASID

Function: BLSQVPAS checks an IKJIDENT PDE containing an ASID to ensure that the ASID is in the range 1 - 32767 or is X 'FFFA'.

Modules Called: BLSUALLO, BLSUFREE, BLSQSIGR

Services Invoked: None

Input

1. PDE (macro IKJIDENT)
2. ABDPL (macro BLSABDPL)
3. Dummy message holder

Output: None

Exit Control Table (ECT) Service

The program control flow showing the modules involved in processing the common services is presented in Figure 68.

Note: In the figure, when BLSQROUT is called, the requested common exit service code is indicated in parentheses.

BLSQECT - ECT exit service request BLSQALLO - Allocate automatic storage BLSQROUT(ADPLSPRT) - Print service BLSQROUT(ADPLSNDX) - Index service BLSQFREE - Free automatic storage
--

Figure 68. ECT Services Control Flow

For lower levels of control flow refer to the individual module descriptions.

BLSQECT: ECT Exit Service

Function: BLSQROUT calls BLSQECT, which passes control to one or more exit modules. If a verb name is given, BLSQECT passes control to the load module that corresponds to the verb name in the AMDRPECT. If the verb name is not passed, BLSQECT checks field ADPLPEFG and determines which exits should be passed control according to which bits are on.

Modules Called: BLSQALLO, BLSQFREE, BLSQROUT

Services Invoked: Index and print

Input: Fields in the system router area (BLSQSRA)

1. SRAECTP - address of the ECT
2. SRAECTN - number of entries in the ECT
3. SRAASIZE - size of the automatic area
4. SRAFLAGS - router flags

Fields in the exit parameter list (BLSABDPL)

1. ADPLPEFG - flags for choosing the exit
2. ADPLPEVB - verb on which to perform the search

Output: ADPLPERR - returned processing information

Select ASID Service

The program control flow showing the modules involved in performing the select ASID service is presented in Figure 69 .

Note: In the figure, when BLSQROUT is called, the requested common exit service code is indicated in parentheses.

BLSQSEL - Select ASID Service BLSQALLO - Allocate automatic storage GETMAIN - Obtain storage for work area, ASVT, and output data BLSQROUT(ADPLSACC) - Storage access service BLSQROUT(ADPLSGTS) - Get symbol service BLSQROUT(ADPLSEQS) - Equate symbol service BLSQROUT(ADPLSPRT) - Print service FREEMAIN - Release storage for work area, ASVT, and output data BLSQFREE - Free automatic storage

Figure 69. Select ASID Service Control Flow

For lower levels of control flow refer to the individual module descriptions.

BLSQSEL: Select ASID Service

Function: BLSQSEL is an exit program that other exit programs invoke using BLSQROUT. BLSQSEL creates a list of ASIDs in the current dump that meets the criteria of the selection keywords specified in the input character string. BLSQSEL also creates symbols for the major control blocks accessed during processing.

Modules Called: BLSQALLO, BLSQFREE, BLSQROUT

Services Invoked: Equate symbol, get symbol, print and storage access

Input

1. Exit parameter list BLSABDPL
2. Select ASID input parameter list

Output: List of selected ASIDs

Data Areas Used: None

Format Service

The program control flow showing the modules involved in performing the format service is presented in Figure 70 .

Note: In the figure, when BLSQROUT is called, the requested common exit service code is indicated in parentheses.

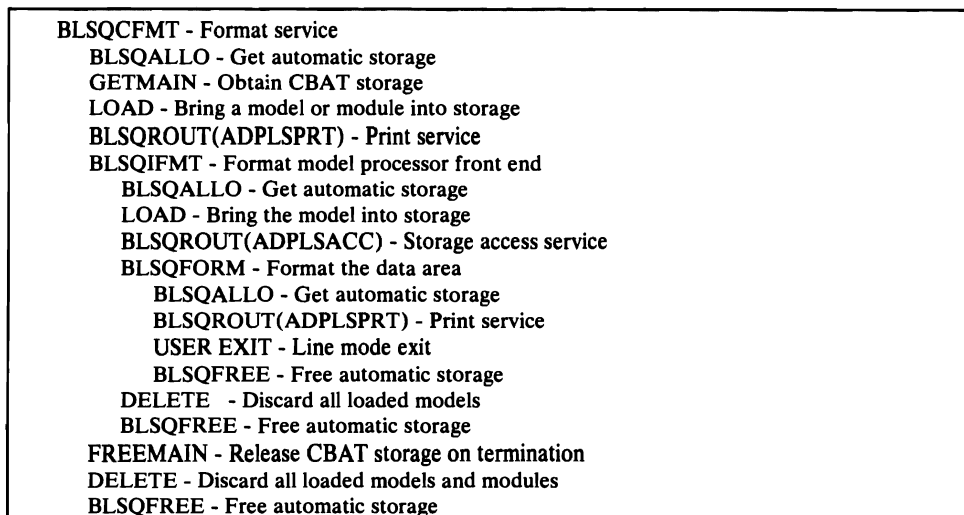


Figure 70. Format Service Control Flow

For lower levels of control flow refer to the individual module descriptions.

BLSQCFMT: Format Service

Function: Exit programs invoke BLSQCFMT via BLSQROUT. BLSQCFMT is itself an exit program. BLSQCFMT performs the following: looks up in the control block acronym table (CBAT) the model and program associated with the control block acronym specified by the caller; if necessary, accesses the control block from the dump; and if a program was loaded, gives the program control. Otherwise, BLSQIFMT is called to format the control block. BLSQIFMT may be called directly by an exit program via the BLSQROUT, with the same input data. However, the caller must specify a model by name or location.

Modules Called: BLSQALLO, BLSQFREE, BLSQIFMT, BLSQROUT

Services Invoked: Print and storage access

Input

1. Exit parameter list BLSABDPL
2. Format parameter list ADPLPFMT

Output: Updated ADPLPFMT

Data Areas Used: BLSQSRA

BLSQIFMT: Format Model Processor

Function: BLSQIFMT ensures that the control block model and the control block are available in storage, processes the index and acronym checking options, and calls the formatting service, BLSQFORM.

Modules Called: BLSQALLO, BLSQFORM, BLSQFREE, BLSQROUT

Services Invoked: Print and storage access

Input

1. BLSABDPL - exit parameter list
2. Updated format model processor parameter list (ADPLPFMT)

Output: ADPLPRET - return processing information

BLSQFORM: Format Model Processor

Function: BLSQIFMT calls BLSQFORM to format and print a data area that is accessed from a dump.

Modules Called: BLSQALLO, BLSQFREE, BLSQROUT, EXITRTN

Services Invoked: Print

Input

1. BLSABDPL - exit parameter list
2. ADPLPFMT - format service parameter list

Output: None

Get Symbol Service

The program control flow showing the modules involved in processing the get symbol service is presented in Figure 71 .

BLSQESGU - Get symbol exit service request BLSUALLO - Allocate automatic storage BLSRESGU - Retrieves the symbol entry from the symbol table BLSUFRE1 - Free automatic storage

Figure 71. Get Symbol Service Control Flow

For lower levels of control flow refer to the individual module descriptions.

BLSQESGU: Get Symbol

Function: BLSQROUT calls BLSQESGU to obtain the definition of an IPCS symbol. If no definition can be obtained, BLSRESGU transmits message BLS18104I.

Modules Called: BLSRESGU, BLSUALLO, BLSUFRE1

Services Invoked: None

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A buffer for an equate symbol record (BLSRESSY). Field ESPASYM must be initialized. If the optional dynamic resolution service is requested, field ESPADT is initialized.

Output: The buffer contains the requested equate symbol record.

Equate Symbol Service

The program control flow showing the modules involved in processing the equate symbol service is presented in Figure 72 .

BLSQESAR - Equate symbol exit service request BLSUALLO - Allocate automatic storage BLSRESAR - Stores the symbol entry into the symbol table BLSUFRE1 - Free automatic storage

Figure 72. Equate Symbol Service Control Flow

For lower levels of control flow refer to the individual module descriptions.

BLSQESAR: Equate Symbol

Function: BLSQROUT calls BLSQESAR to store a record in the symbol table. If a symbol entry already exists, BLSQESAR overlays the entry.

Modules Called: BLSRESAR, BLSUALLO, BLSUFRE1

Services Invoked: None

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. An equate symbol record (macro BLSRESSY) that is completely initialized except for field ESBARDX.

Output: Field ESBARDX contains the current dump reference number.

Line-Oriented Dump Viewing Subcommands

The dump viewing subcommands are used to find various portions of the dump and run through chains of control blocks. The output generated by these subcommands is transmitted by the TSO I/O service routines and the standard TSO terminal-independent interfaces, permitting use from a variety of TSO terminals.

The program control flow charts for the line-oriented dump viewing subcommands are shown in Figure 73 through Figure 78 on page 220 .

FIND Subcommand

The program control flow chart showing the modules involved in processing the FIND subcommand is presented in Figure 73 .

BLSRFIND - IPCS FIND main processor
BLSUALLO - Allocate automatic storage
BLSUPARI - IKJPARS interface
BLSRVPAD - Validate PDE for address
BLSRVPAS - Parse validity check for ASID ident PDE
BLSRVPCP - Parse validity check for CPU ident.
BLSRVPVA - Parse validity check value PDE
BLSUVP31 - Validate for integer 1 through 2^{31}
BLSUVP32 - Validate for integer 0 through $2^{31}-1$
BLSUVP33 - Validate for integer 0 through $2^{24}-1$
BLSUPARU - Process routing operands
BLSRPADS - Process display operands
BLSRDUSO - Prepare dump for use
BLSUPGME - Call non-resident program
BLSRVAL - Value resolution router
BLSRSDGE - Get session default record
BLSUMPKN - Compress text of message
BLSUTRMV - Transmit variable-length message
BLSUPGME - Call non-resident program
BLSRM214 - Diagnose invalid MASK
BLSUPGME - Call non-resident program
BLSRSDAR - Add/Replace SD record
BLSRADDS - Resolve address PDEs
BLSRESGE - Retrieve equate symbol definition
BLSUPUTV - Route variable-length message
BLSRESAR - Add or replace an equate symbol record
BLSRRAGE - Retrieve address space description
BLSRACCQ - Access dump storage
BLSRACCA - Access dump record
BLSRACCT - Access dump record
BLSRESAR - Add or replace an equate symbol record
BLSUPGME - Call non-resident program
BLSRFISE - Diagnose invalid MASK
BLST01 - Display unbuffered storage routine
BLSUFRE1 - Free automatic storage

Figure 73. FIND Subcommand Control Flow

For lower levels of control flow, refer to the individual module descriptions.

BLSRFIND: FIND Subcommand Processor

Function: BLSRFIND finds a given hex or character string in the dump within a given search range, at a given boundary alignment, with optional masking of insignificant bits of the search argument. If NOBREAK is specified, RA records are retrieved to find the next available data in storage when a discontinuity is found. If neither a search argument nor an address range is given, the search starts at X+1. If only the address range is omitted, the search starts at X.

Modules Called: BLSRADDs, BLSRDUSO, BLSRESAR, BLSRESGE, BLSRFISE, BLSRM214, BLSRPADS, BLSRSDGE, BLSRVAL, BLSRVPAD, BLSRVPAS, BLSRVPCP, BLSRVPA, BLST01, BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUPARI, BLSUPARU, BLSUPGME, BLSUTRMV, BLSUVPVA, BLSUVP31, BLSUVP32, BLSUVP33

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

Value Resolution Service

BLSRVAL: Value Resolution Service

Function: BLSRVAL calls the following modules to perform the translations from the input formats indicated.

Module	Input Format
BLSRVALA	Fullword pointer
BLSRVALC	Character string
BLSRVALF	Fullword signed
BLSRVALH	Halfword signed
BLSRVALP	Picture string
BLSRVALT ²	Text string
BLSRVALX	Hexadecimal string

If the data cannot be resolved, IPCS processing issues a diagnostic message to the terminal and an informative return code to the caller.

Modules Called: BLSRVALA, BLSRVALC, BLSRVALF, BLSRVALH, BLSRVALP, BLSRVALT, BLSRVALX, BLSUALLO, BLSUFRE1, BLSUPGMC, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Pointer to the PDE to be processed
3. Pointer to a 256 character buffer

Output: The translated value is placed in the buffer.

BLSRVALA: Resolve Pointer Value

Function: BLSRVALA translates data described by a value in the PDE (generated by IKJPARS) from a fullword pointer format to binary and returns the result to the caller in the buffer provided. If the data cannot be resolved, IPCS processing issues a diagnostic message to the terminal and an informative return code to the caller.

Modules Called: BLSRM192, BLSRM193, BLSUALLO, BLSUFRE1, BLSUPGME

² BLSRVALT is a secondary entry point in BLSRVALC. It is not a separate module.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Pointer to the PDE to be processed
3. Pointer to a 256 character buffer

Output: The translated value is placed in the buffer.

Data Areas Used: None

BLSRVALC: EBCDIC Value Resolution

Function: BLSRVALC translates data described by a value in the PDE (generated by IKJPARS) from a character string format to binary. BLSRVALT is a secondary entry point within BLSRVALC that performs an additional translation of data from a text string format to binary. BLSRVALC returns the result to the caller in the buffer provided. If the data cannot be resolved, IPCS processing issues a diagnostic message to the terminal and an informative return code to the caller.

Modules Called: BLSUALLO, BLSUFRE1, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Pointer to the PDE to be processed
3. Pointer to a 256 character buffer

Output: The translated value is placed in the buffer.

Data Areas Used: None

BLSRVALF: Resolve Numeric Value

Function: BLSRVALF translates data described by a value in the PDE (generated by IKJPARS) from a fullword signed format to binary and returns the result to the caller in the buffer provided. If the data cannot be resolved, IPCS processing issues a diagnostic message to the terminal and an informative return code to the caller.

Modules Called: BLSRM192, BLSRM193, BLSRM199, BLSRM200, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Pointer to the PDE to be processed
3. Pointer to a 256 character buffer

Output: The translated value is placed in the buffer.

Data Areas Used: None

BLSRVALH: Resolve Numeric Value

Function: BLSRVALH translates data described by a value in the PDE (generated by IKJPARS) from a halfword signed format to binary and returns the result to the caller in the buffer provided. If the data cannot be resolved, IPCS processing issues a diagnostic message to the terminal and an informative return code to the caller.

Modules Called: BLSRM192, BLSRM193, BLSRM199, BLSRM200, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Pointer to the PDE to be processed
3. Pointer to a 256 character buffer

Output: The translated value is placed in the buffer.

Data Areas Used: None

BLSRVALP: Resolve Picture String

Function: BLSRVALP translates data described by a value in the PDE (generated by IKJPARS) from a picture string format to binary and returns the result to the caller in the buffer provided. If the data cannot be resolved, IPCS processing issues a diagnostic message to the terminal and an informative return code to the caller.

Modules Called: BLSUALLO, BLSUFRE1, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Pointer to the PDE to be processed
3. Pointer to a 256 character buffer

Output: The translated value is placed in the buffer.

Data Areas Used: None

BLSRVALX: Hexadecimal Value Resolution

Function: BLSRVALX translates data described by a value in the PDE (generated by IKJPARS) from a hexadecimal string format to binary and returns the result to the caller in the buffer provided. If the data cannot be resolved, IPCS processing issues a diagnostic message to the terminal and an informative return code to the caller.

Modules Called: BLSRM192, BLSRM193, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Pointer to the PDE to be processed
3. Pointer to a 256 character buffer

Output: The translated value is placed in the buffer.

Data Areas Used: None

FINDMOD Subcommand

The program control flow showing the modules involved in processing the FINDMOD subcommand is presented in Figure 74 .

BLSRFMOD - IPCS FINDMOD main processor
BLSUALLO - Allocate automatic storage
BLSUPARI - IKJPARS interface
BLSUPARU - Process routing operands
BLSRPADS - Process display operands
BLSRDUSO - Prepare dump for use
BLSUPUTV - Route variable-length message
BLSRESGE - Retrieve equate symbol record definition
BLSRESGU - Retrieve active equate symbol record
BLSRACC - Debugging storage access service routine
BLST01 - Display unbuffered storage routine
BLSRESAR - Add or replace an equate symbol record
BLSUPUTO - Route single-level message
BLSUPUTV - Route variable-length message
BLSUFRE1 - Free automatic storage

Figure 74. FINDMOD Subcommand Control Flow

For lower levels of control flow (for example, calls to common service routines), refer to the individual module descriptions.

***BLSRFMOD:* FINDMOD Subcommand Processor**

Function: BLSRFMOD finds a given module, which may be expressed in character or hex (for example, SVCs with overpunch characters). The search goes first to the equate symbol records (symbol table). If the symbol requested is entered there and the type field specifies module, that value is returned. If no symbol is found, the CDE chain for the active LPA is searched. If no CDE is found for the module name, the link pack directory is searched to locate the module. Equate symbol records are produced for any modules, CDEs, XLs, or LPDEs. A message is produced for minor LPDEs or CDEs.

Modules Called: BLSRACC, BLSRDUSO, BLSRESAR, BLSRESGE, BLSRESGU, BLSRPADS, BLST01, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPARU, BLSUPUTO, BLSUPUTV

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

Data Areas Used: CVT, CDE, XTLST, LPDE

FINDUCB Subcommand

The program control flow showing the modules involved in processing the FINDUCB subcommand is presented in Figure 75 .

BLSRFUCB - IPCS FINDUCB main processor
BLSUALLO - Allocate automatic storage
BLSUPARI - IKJPARS interface
BLSUVPX1 - Parse validity check exit for IKJIDENT PDE
BLSUPARU - Process routing operands
BLSRPADS - Process display options
BLSRDUSO - Prepare dump for use
BLSRESGU - Retrieve active equate symbol record
BLSRESAR - Add or replace an equate symbol record
BLST01 - Display unbuffered storage routine
BLSUFRE1 - Free automatic storage

Figure 75. FINDUCB Subcommand Control Flow

For lower levels of control flow, refer to the individual module descriptions.

BLSRFUCB: FINDUCB Subcommand Processor

Function: BLSRFUCB calls a routine that locates a given unit control block (UCB). BLSRFUCB processing sets the symbol X to the UCB address, creates an equate symbol record for the name UCBnnn (where nnn is the device number), and optionally, displays the UCB.

Modules Called: BLSRDUSO, BLSRESAR, BLSRESGU, BLSRPADS, BLST01, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPARU, BLSUVPX1

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

LIST Subcommand

The program control flow showing the modules involved in processing the LIST subcommand is presented in Figure 76 .

BLSRLIST - IPCS LIST main processor
BLSUALLO - Allocate automatic storage
BLSUPARI - IKJPARS interface
BLSUVP31 - Validate for integer 1 through 2 ³¹
BLSUVP32 - Validate for integer 0 through 2 ³¹ -1
BLSUVP33 - Validate for integer 0 through 2 ²⁴ -1
BLSRVPCP - Parse validity check exit for CPU ident PDE
BLSRVPAS - Parse validity check for ASID ident PDE
BLSRVPAD - Address validity check subroutine
BLSUPARU - Process routing operands
BLSRPADS - Process display operands
BLSRADDs - Resolve address PDEs
BLSRADDR - Resolve address PDEs
BLST01 - Display unbuffered storage routine
BLSUFRE1 - Free automatic storage

Figure 76. LIST Subcommand Control Flow

For lower levels of control flow, refer to the individual module descriptions.

BLSRLIST: LIST Subcommand Processor

Function: BLSRLIST parses an address expression or list of address expressions. BLSRLIST then calls a routine to display the resultant address contents.

Modules Called: BLSRADDR, BLSRADD5, BLSRPADS, BLSRVPAD, BLSRVPAS, BLSRVPCP, BLST01, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPARU, BLSUVP31, BLSUVP32, BLSUVP33

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

LISTUCB Subcommand

The program control flow showing the modules involved in processing the LISTUCB subcommand is presented in Figure 77 .

BLSRLUCB - LISTUCB subcommand processor
BLSUALLO - Allocate automatic storage
BLSUPARI - Resolve address PDEs
BLSUVPX2 - Validate PDE, hex range
BLSUPARU - Process routing
BLSRPADS - Process display options
BLSRDUSO - Prepare dump for use
BLSRESGU - Get ES record
BLSRESAR - Store ES record, update X
BLST01 - Display one UCB
BLSUFRE1 - Free automatic storage

Figure 77. LISTUCB Subcommand Control Flow

For lower levels of control flow, refer to the individual module descriptions.

BLSRLUCB: LISTUCB Subcommand Processor

Function: BLSRLUCB locates the specified UCBs in a dump and displays them, one at a time. BLSRLUCB processing also sets the symbol X to the address of each UCB as it is processed and creates an equate symbol record for each UCB processed.

Modules Called: BLSRDUSO, BLSRESAR, BLSRESGU, BLSRPADS, BLST01, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPARU, BLSUVPX2

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

RUNCHAIN Subcommand

The program control flow showing the modules involved in processing the RUNCHAIN subcommand is presented in Figure 78 on page 220 .

BLSRRNCH - IPCS RUNCHAIN main processor
 BLSUALLO - Allocate automatic storage
 BLSUPARI - IKJPARS interface
 BLSUVP31 - Validate for integer 1 through 2³¹
 BLSUVP32 - Validate for integer 0 through 2³¹-1
 BLSUVP33 - Validate for integer 0 through 2²⁴-1
 BLSRVPAD - Validate PDE for address
 BLSRVPAS - Validate PDE for ADID
 BLSRVPCP - Validate PDE for CPU address
 BLSUPARU - Process routing operands
 BLSRPADS - Process display operands
 BLSRADDs - Resolve address PDEs
 BLSUSTKT - STack terminal element
 BLSUPUTV - Transmit BLS18150I
 BLSRESSR - Scan data area
 BLSUPUTV - Get storage address record unconditionally
 BLSRSSSA - Substitute standard name for alias
 BLSRESAR - Add or replace an equate symbol record
 BLST01 - Display unbuffered storage routine
 BLSUSTKS - Stack subcommand/CLIST
 BLSUMON - Process subcommand/CLIST
 BLSRDUCK - Prepare dump for use
 BLSRESCK - Check equate symbol record
 BLSRACC - Debugging storage access routine
 BLSUMPK1 - Compress a message
 BLSUPUTO - Route single-level message
 BLSUSTKD - Delete terminal element
 BLSUFRE1 - Free automatic storage

Figure 78. RUNCHAIN Subcommand Control Flow

For lower levels of control flow, refer to the individual module descriptions.

BLSRRNCH: RUNCHAIN Subcommand Processor

Function: BLSRRNCH accesses and follows a chain of control blocks in the dump, optionally naming them and entering the names in the symbol table. The user specifies the beginning address, length, etc. of the first block, the maximum number of blocks to chain, the offset and mask for the forward pointer, the value of the forward pointer (denoting the end of the chain), and the display options for the blocks.

Modules Called: BLSRACC, BLSRADDs, BLSRDUCK, BLSRESAR, BLSRESCK, BLSRESSR, BLSRPADS, BLSRSSSA, BLSRVPAD, BLSRVPAS, BLSRVPCP, BLST01, BLSUALLO, BLSUFRE1, BLSUMON, BLSUMPK1, BLSUPARI, BLSUPARU, BLSUPUTV, BLSUSTKD, BLSUSTKS, BLSUSTKT, BLSUVP31, BLSUVP32, BLSUVP33

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

Full-screen Dump Viewing (DSPL3270)

The program control flow showing the modules involved in processing the DSPL3270 subcommand is presented in Figure 79 on page 221 .

BLSR3270 - DSPL3270 subcommand processor
BLSUALLO - Allocate automatic storage
BLSUPARI - Parse subcommand operands
BLSRDUSO - Prepare dump for use
BLSUPGMR - Call non-resident program
BLSRPRXC - Create exit interface
BLSUTRMV - Transmit error messages (if any)
BLSRESGU - Retrieve system CVT
BLSRESGE - Retrieve IPCS current symbol (X)
BLSRESAR - Establish new current symbol definition
BLSUPGMC - Call non-resident program
BLSRPHGE - Read DSPL3270 history record
BLSR327A - Internal procedure
BLSRACCL - Acquire dump data for display
BLSRVPAS - Validate user-supplied ASID
IKJSTCK - Add subcommand to TSO input stack
BLSR3270 - Resume mainline
BLSRESAR - Update current symbol definition
BLSUPGMC - Call non-resident program
BLSRPHAR - Update DSPL3270 history record
BLSUFRE1 - Free automatic storage

Figure 79. DSPL3270 Subcommand Control Flow

For lower levels of control flow (for example, calls to common service routines), refer to the individual module descriptions.

BLSR3270: IPCS DSPL3270 Subcommand Processor

Function: DSPL3270 provides a full-screen dump viewing capability. Control of the display screen is gained by notifying TSO TIOC (TCAM/VTAM) that full-screen mode is being entered. While in control of the screen, DSPL3270 performs all terminal input (TGET) and output (TPUT), including the construction of a formatted output buffer. Support for cursor tabbing, recognition of cursor position, and the use of predefined program function keys is provided. Requests for the display of dump data are validity checked prior to display image regeneration. At termination, control of the display screen is relinquished by notifying TSO that full-screen mode is being exited and resetting the current screen line number. The majority of processing is performed by CSECT BLSR327A.

Modules Called: BLSRACCL, BLSRDUSO, BLSRESAR, BLSRESGE, BLSRESGU, BLSRPHAR, BLSRPHGE, BLSRPRXC, BLSRVPAS, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPGMC, BLSUPGMR, BLSUTRMV, IKJSTCK

Input: Active task variable (see “BLSUZZ2” on page 406)

Output

1. The IPCS current symbol (X) is updated to reflect the area of storage last referenced within DSPL3270.
2. The DSPL3270 program history (PH) records are updated.

Data Areas Used: CPPL, ECT, IOPL, LSD, UPT

Common Services for CLIST Support

The following modules provide support for the CLIST support subcommands.

BLSUVAST: Update External Variable

Function: BLSUVAST updates a variable maintained outside IPCS by:

- TSO CLIST support.
- ISPF dialog support.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUTRMV, BLSUVARX, IKJCT441, ISPLINK

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Type of variable
3. Variable name
4. Update value

Output: None

CLIST Support Subcommands

The CLIST support subcommands enhance CLIST execution control flow, and provide message production and pagination control for the print data set. The program control flow charts for the CLIST support subcommands are shown in Figure 80 on page 223 through Figure 88 on page 229 .

COMPARE Subcommand

The program control flow showing the modules involved in processing the COMPARE subcommand is presented in Figure 80 on page 223 .

BLSRCOMP - IPCS COMPARE main processor
 BLSUALLO - Allocate automatic storage
 BLSUPARI - IKJPARS interface
 BLSUVP31 - Validate for integer 1 through 2^{31}
 BLSUVP32 - Validate for integer 0 through $2^{31}-1$
 BLSUVP33 - Validate for integer 0 through $2^{24}-1$
 BLSRVPCP - Parse validity check exit for CPU ident
 BLSRVPAS - Parse validity check for ASID ident PDE
 BLSRVPAD - Address validity check subroutine
 BLSRVPVA - Parse validity check exit for value PDE
 BLSUPGME - Call non-resident IPCS program
 BLSRM214 - Transmit message BLS18214I
 BLSUPARU - Process routing operands
 BLSRPADS - Process display operands
 BLSUDDSN - Set ZZ2AD
 BLSRADDR - Resolve address PDEs
 BLSUPGME - Call non-resident program
 BLSRVAL - Value resolution router
 BLSUPUTV - Route variable-length message
 BLSRDUSO - Prepare dump for use
 BLSRACC - Debugging storage access service routine
 BLSUPGME - Call non-resident IPCS program
 BLSRCPIC - Compare picture strings
 BLSUMPK1 - Compress a message
 BLSUTRMV - Transmit message to terminal
 BLSUPUTO - Route single-level message
 BLSUFRE1 - Free automatic storage

Figure 80. COMPARE Subcommand Control Flow

For lower levels of control flow, refer to the individual module descriptions.

BLSRCOMP: COMPARE Subcommand Processor

Function: BLSRCOMP performs a logical comparison between a literal value and dump data, value versus value, data versus data, or data versus value. A message is optionally produced giving the results, but the primary use of the subcommand is in CLISTs, where the return code may be analyzed.

Modules Called: BLSRACC, BLSRADDR, BLSRCPIC, BLSRDUSO, BLSRM214, BLSRPADS, BLSRVPA, BLSRVPCP, BLSRVVPA, BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUPARI, BLSUPARU, BLSUPGME, BLSUPUTO, BLSUPUTV, BLSUTRMV, BLSUVPVA, BLSUVP31, BLSUVP32, BLSUVP33

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSRCPIC: COMPARE Picture Service Routine

Function: BLSRCPIC performs a logical comparison between a picture string and data.

Modules Called: None

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. General value buffer (see “BLSRVALY” on page 396)
3. Data buffer

Output: None

EVALDEF Subcommand

The program control flow chart for the EVALDEF subcommand is shown in Figure 81.

BLSREDEF - EVALDEF subcommand
BLSUALLO - Allocate automatic storage
BLSUPARI - IKJPARS interface
BLSUDDSN - Locate ZZ6 for default DSNAME
BLSUPGME - Call non-resident program
BLSRRDGE - Get record
BLSRMSGGA - Format address space
BLSUMPKN - Compress message
BLSRMDSN - Format DSNAME
BLSUMPK1 - Compress message
BLSUTRMV - Transmit message
BLSUVAST - Update variable
BLSUFRE1 - Free automatic storage

Figure 81. EVALDEF Subcommand Control Flow

For lower levels of control flow (for example calls to common service routines), refer to the individual module descriptions.

***BLSREDEF:* EVALDEF Subcommand Processor**

Function: Store IPCS default values in one or more variables maintained outside IPCS.

Modules Called: BLSRMDSN, BLSRMSGGA, BLSRRDGE, BLSUALLO, BLSUDDSN, BLSUFRE1, BLSUMPKN, BLSUMPK1, BLSUPARI, BLSUPGME, BLSUTRMV, BLSUVAST

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

EVALDUMP Subcommand

The program control flow chart for the EVALDUMP subcommand is shown in Figure 82 on page 225.

BLSREDUM - EVALDUMP subcommand
BLSUALLO - Allocate automatic storage
BLSUPARI - IKJPARS interface
BLSUDUSE - Record the specified dump
BLSUVSCR - Create RPL
BLSUVSPO - Sequential get origin
BLSUVSGU - Sequential get
BLSUVSEN - VSAM ENDREQ
BLSUMDSN - Format the dump source
BLSRMSGGA - Format address space
BLSUMPKN - Compress message text
BLSUVAST - Update external variable
BLSUFRE1 - Free automatic storage

Figure 82. EVALDUMP Subcommand Control Flow

For lower levels of control flow (for example calls to common service routines), refer to the individual module descriptions.

BLSREDUM: EVALDUMP Subcommand Processor

Function: BLSREDUM retrieves the description of a designated dump. If the dump is described, one or more variables may be updated to reflect the description retrieved.

Modules Called: BLSRDUSE, BLSRMDSN, BLSRMSGGA, BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUPARI, BLSUVAST, BLSUVSCR, BLSUVSEN, BLSUVSGU, BLSUVSPO

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

EVALMAP Subcommand

The program control flow chart for the EVALMAP subcommand is shown in Figure 83 on page 226.

BLSREMAP - EVALMAP subcommand
 BLSUALLO - Allocate automatic storage
 BLSUPARI - IKJPARS interface
 BLSUVP31 - Validate for integer 1 through 2^{31}
 BLSUVP32 - Validate for integer 0 through $2^{31}-1$
 BLSUVP33 - Validate for integer 0 through $2^{24}-1$
 BLSRVPAD - Validate PDE for address
 BLSRVPAS - Validate PDE for ASID
 BLSRVPCP - Validate PDE for CPU address
 BLSRADDS - Resolve address and update X
 BLSRDUSO - Prepare dump for use
 BLSUPARU - Process routing operands
 BLSRSAGN - Determine group data type
 BLSUVSCR - Create RPL
 BLSUVSPO - Sequential get origin
 BLSUVSGU - Sequential get
 BLSUVSEN - VSAM ENDREQ
 BLSRMSGGA - Format address space
 BLSRMSGD - Format data type
 BLSUMPKN - Compress message
 BLSUMPK1 - Compress message
 BLSUVAST - Update external variable
 BLSUFRE1 - Free automatic storage

Figure 83. EVALMAP Subcommand Control Flow

BLSREMAP: EVALMAP Subcommand Processor

Function: BLSREMAP retrieves the definition of a designated map. One or more CLIST variables may be updated to reflect the description retrieved.

Modules Called: BLSRADDS, BLSRDUSO, BLSRMSGGA, BLSRMSGD, BLSRSAGN, BLSRVPAD, BLSRVPAS, BLSRVPCP, BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUMPK1, BLSUPARI, BLSUPARU, BLSUVAST, BLSUVP31, BLSUVP32, BLSUVP33, BLSUVSCR, BLSUVSEN, BLSUVSGU, BLSUVSPO

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

EVALSYM Subcommand

The program control flow chart for the EVALSYM subcommand is shown in Figure 84 on page 227.

BLSRESYM - EVALSYM subcommand
 BLSUALLO - Allocate automatic storage
 BLSUPARI - IKJPARS interface
 BLSRDUSO - Prepare dump for use
 BLSRSSA - Alias to standard symbol
 BLSUVSCR - Create RPL
 BLSUVSPO - Sequential get origin
 BLSUVSGU - Sequential get
 BLSUVSEN - VSAM ENDREQ
 BLSUPGME - Call non-resident program
 BLSRECHA - Format character data
 BLSRSSC - Compress standard symbol
 BLSRMSG - Format qualifiers
 BLSRMSGD - Format data type
 BLSUMPKN - Compress message
 BLSUVAST - Store external variable
 BLSUFRE1 - Free automatic storage

Figure 84. EVALSYM Subcommand Control Flow

For lower levels of control flow (for example calls to common service routines), refer to the individual module descriptions.

BLSRESYM: EVALSYM Subcommand Processor

Function: BLSRESYM retrieves the definition of a designated symbol. One or more CLIST variables may be updated to reflect the description retrieved.

Modules Called: BLSRDUSO, BLSRECHA, BLSRMSG, BLSRMSGD, BLSRSSA, BLSRSSC, BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUPARI, BLSUPGME, BLSUVAST, BLSUVSCR, BLSUVSEN, BLSUVSGU, BLSUVSPO

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

EVALUATE Subcommand

The program control flow chart showing the modules involved in processing the EVALUATE subcommand is shown in Figure 85.

BLSREVAL - IPCS EVALUATE main processor
 BLSUALLO - Allocate automatic storage
 BLSUPARI - IKJPARS interface
 BLSUVP31 - Validate for integer 1 through 2^{31}
 BLSUVP32 - Validate for integer 0 through $2^{31}-1$
 BLSUVP33 - Validate for integer 0 through $2^{24}-1$
 BLSRVPCP - Parse validity check exit for CPU ident
 BLSRVPAS - Parse validity check for ASID ident PDE
 BLSRVPAD - Address validity check subroutine
 BLSRVPCP - Validate CPU address
 BLSUPARU - Process routing operands
 BLSUPGME - Call non-resident program
 BLSRECHA - Format character data
 BLSUMPKN - Compress message
 BLSRADDR - Resolve address PDEs
 BLSRESAR - Add/replace equate symbol record
 BLSRACC - Debugging storage access service routine
 BLSUVAST - Update external variable
 BLSUFRE1 - Free automatic storage

Figure 85. EVALUATE Subcommand Control Flow

BLSREVAL: EVALUATE Subcommand Processor

Function: BLSREVAL retrieves one to four bytes of storage as specified by the address operands and places the data, right-adjusted, in register 15. The first byte of register 15 is always set to zero, and additional leading zeroes are supplied if one or two bytes are requested.

Modules Called: BLSRACC, BLSRADDR, BLSRECHA, BLSRESAR, BLSRVPAD, BLSRVPAS, BLSRVPCP, BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUPARI, BLSUPARU, BLSUPGME, BLSUVAST, BLSUVP31, BLSUVP32, BLSUVP33

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

INTEGER Subcommand

The program control flow chart for the INTEGER subcommand is shown in Figure 86.

BLSULIN - INTEGER subcommand
BLSUALLO - Allocate automatic storage
BLSUPARI - IKJPARS interface
BLSUVPS1 - Validate PDE for one signed integer
BLSUPARU - Process routing options
BLSUSIGR - Process signed integer range
BLSUPGME - Call non-resident program
BLSRECHA - Format character data
BLSUPUTD - Transmit message
BLSUVAST - Store external variable
BLSUFRE1 - Free automatic storage

Figure 86. INTEGER Subcommand Control Flow

For lower levels of control flow (for example calls to common service routines), refer to the individual module descriptions.

BLSULIN: INTEGER Subcommand Processor

Function: Format a fullword.

Modules Called: BLSRECHA, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPARU, BLSUPGME, BLSUPUTD, BLSUSIGR, BLSUVAST, BLSUVPS1

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

ISPEXEC Subcommand

The program control flow showing the modules involved in processing the ISPEXEC subcommand is presented in Figure 87 on page 229

BLSGX - Process ISPEXEC subcommand BLSUALLO - Allocate automatic storage BLSUTRMV - Transmit terminal message BLSUPGMU - Load ISPEXEC service routine IKJCT441 - Check for an active CLIST ISPLINK - Establish VDEFINE exit ISPEXEC - Perform SPF service BLSUPGMD - Delete ISPEXEC service routine BLSUFRE1 - Free automatic storage

Figure 87. ISPEXEC Processing Control Flow

For lower levels of control flow, refer to the individual module descriptions.

BLSGX: ISPEXEC Subcommand Processor

Function: BLSGX processes the ISPEXEC subcommand that is issued from a CLIST or that is entered on a dialog panel. BLSGX passes the command to the ISPEXEC service routine to parse the operands and to have the service performed.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPGMD, BLSUPGMU, BLSUTRMV, IKJCT441, ISPEXEC, ISPLINK

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

NOTE Subcommand

The program control flow chart showing the modules involved in processing the NOTE subcommand is presented in Figure 88 .

BLSUNOTE - IPCS NOTE main processor BLSUALLO - Allocate automatic storage BLSUPARI - IKJPARS interface BLSUPARU - Process routing operands BLSUPUTA - Route variable-length message. BLSUFRE1 - Free automatic storage

Figure 88. NOTE Subcommand Control Flow

For lower levels of control flow, refer to the individual module descriptions.

BLSUNOTE: NOTE Subcommand Processor

Function: BLSUNOTE satisfies output paging and spacing control needs. It also formats and transmits user-supplied text to the active IPCS output destination(s).

Modules Called: BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPARU, BLSUPUTA

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

Dump Directory Handling Subcommands

The dump directory subcommands manage the dump data cross-reference records, symbol table records, and storage mapping records.

OPEN Subcommand

The program control flow chart for the OPEN subcommand is shown in Figure 89 .

BLSUOPEN - Open dump data set/print file
BLSUALLO - Allocate automatic storage
BLSUPARI - IKJPARS interface
BLSUPGMU - Load RVT
BLSRDUSO - Prepare the dump
BLSRMDSN - Format the dump source
BLSUPGMR - Call non-resident IPCS program
BLSRZZ3U - Update IPCS defaults
BLSUMPKN - Compress the message
BLSUPROP - Open the print file
BLSUTRMV - Transmit message
BLSUFRE1 - Free automatic storage

Figure 89. OPEN Subcommand Control Flow

For lower levels of control flow (for example calls to common service routines), refer to the individual module descriptions.

BLSUOPEN: OPEN Subcommand Processor

Function: BLSUOPEN opens dump data set(s), a print data set, or both.

Modules Called: BLSRDUSO, BLSRMDSN, BLSRZZ3U, BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUPARI, BLSUPGMR, BLSUPGMU, BLSUPROP, BLSUTRMV

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

CLOSE Subcommand

The program control flow chart for the CLOSE subcommand is shown in Figure 90.

BLSUCLOS - Close dump data set/print file
BLSUALLO - Allocate automatic storage
BLSUPARI - IKJPARS interface
BLSUDDSN - Update ZZ2AD
BLSUDUSE - Record the specified dump
BLSRDUCC - Close/free dump DSN
BLSRDUCD - Close/deallocate dump
BLSRMDSN - Format the dump source
BLSRDUCL - Close/free all dumps
BLSRDUCD - Close/deallocate dump
BLSUPRCC - Close print file
BLSUTRMV - Transmit message BLS21075I
BLSUFRE1 - Free automatic storage

Figure 90. CLOSE Subcommand Control Flow

For lower levels of control flow (for example calls to common service routines), refer to the individual module descriptions.

BLSUCLOS: CLOSE Subcommand Processor

Function: BLSUCLOS closes dump data set(s), a print file, or both.

Modules Called: BLSRDUCC, BLSRDUCL, BLSRDUSE, BLSRMDSN, BLSUALLO, BLSUDDSN, BLSUDUCD, BLSUFRE1, BLSUPARI, BLSUPRCC, BLSUTRMV

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

LISTDUMP and DROPDUMP Subcommands

Each module in this group receives control with register 1 addressing the active task variable (see “BLSUZZ2” on page 406) and returns only a return code directly to its caller. Figure 91 summarizes the processing of the four modules in the group.

Module	Function	Modules Called
BLSRDUDE	DROPDUMP subcommand processor	BLSRDUCC, BLSRDUCK, BLSRDUDR, BLSRDUDT, BLSRDUSE, BLSRMDSN, BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUPARI, BLSUPGMS, BLSUTRMV
BLSRDUDR	DROPDUMP RECORDS(ALL) service routine	BLSRRDAR, BLSRRDGE, BLSUALLO, BLSUFRE1, BLSUPGMC, BLSUVSCR, BLSUVSEN, BLSUVSER, BLSUVSGU, BLSUVSPO, BLSUVSWB
BLSRDUDT	DROPDUMP RECORDS(TRANSLATION) service routine	BLSRDUCK, BLSRRAAR, BLSRRDAR, BLSRRDGE, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUVSCR, BLSUVSEN, BLSUVSER, BLSUVSGU, BLSUVSPO, BLSUVSWB
BLSRDULS	LISTDUMP subcommand processor	BLSRDUDR, BLSRMDSN, BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUPARI, BLSUPARU, BLSUPGMS, BLSUPRTT, BLSUPUTA, BLSUTRMA, BLSUVSCR, BLSUVSEN, BLSUVSGU, BLSUVSPO

Figure 91. Summary of LISTDUMP and DROPDUMP

DROPMap, LISTMap, and SCAN Subcommands

The program control flow charts for the DROPMap, LISTMap, and SCAN subcommands are shown in Figure 92 through Figure 94 on page 233 .

BLSRSADE - IPCS DROPMap main processor
BLSUALLO - Allocate automatic storage
BLSUVSCR - Create a request parameter list
BLSUPARI - IKJPARS interface
BLSRVPAD - Address validity check subroutine
BLSRVPAS - Parse validity check for ASID ident PDE
BLSRVPCP - Parse validity check exit for CPU ident
BLSUVP31 - Validate for integer 1 through 2^{31}
BLSUVP32 - Validate for integer 0 through $2^{31}-1$
BLSUVP33 - Validate for integer 0 through $2^{24}-1$
BLSUPARU - Process routing operands
BLSRDUSO - Prepare the dump
BLSRADDR - Resolve address PDEs
BLSRADD2 - Generate address space information
BLSUVSPO - Point to desired record
BLSUVSGU - Get a record from the dump directory
BLSUVSER - Erase record from the dump directory
BLSUVSEN - Terminate a VSAM request string
BLSUPGMC - Call non-resident IPCS program
BLSRRDAR - Update RD record
BLSUMPK1 - Compress a message
BLSUTRMV - Transmit variable-length message to terminal
BLSUFRE1 - Free automatic storage

Figure 92. DROPMap Subcommand Control Flow

BLSRSALI - IPCS LISTMap main processor
BLSUALLO - Allocate automatic storage
BLSUVSCR - Create a request parameter list
BLSUPARI - IKJPARS interface
BLSRVPAD - Address validity check subroutine
BLSRVPAS - Parse validity check for ASID ident PDE
BLSRVPCP - Parse validity check exit for CPU ident
BLSUVP31 - Validate for integer 1 through 2^{31}
BLSUVP32 - Validate for integer 0 through $2^{31}-1$
BLSUVP33 - Validate for integer 0 through $2^{24}-1$
BLSUPARU - Process routing operands
BLSRPADS - Process display operands
BLSRDUSO - Prepare the dump
BLSRADDR - Resolve address PDEs
BLSRADD2 - Generate address space information
BLSUVSPO - Point to desired record
BLSUVSGU - Get a record from the dump directory
BLSUPGMC - Call non-resident IPCS program
BLSVxxxx - Rescan a block
BLSUVSEN - Terminate a VSAM request string
BLSRDATK - Edit storage description
BLST01 - Display unbuffered storage
BLSUMPK1 - Compress a message
BLSUPUTV - Route variable length message
BLSUFRE1 - Free automatic storage

Figure 93. LISTMap Subcommand Control Flow

BLSRSCAN - IPCS SCAN main processor
 BLSUALLO - Allocate automatic storage
 BLSUVSCR - Create a request parameter list
 BLSUPARI - IKJPARS interface
 BLSRVPAD - Address validity check subroutine
 BLSRVPAS - Parse validity check for ASID ident PDE
 BLSRVPCP - Parse validity check exit for CPU ident
 BLSUVP21 - Validate for integer 1 through 65,535
 BLSUVP31 - Validate for integer 1 through 2^{31}
 BLSUVP32 - Validate for integer 0 through $2^{31}-1$
 BLSUVP33 - Validate for integer 0 through $2^{24}-1$
 BLSUPARU - Process routing operands
 BLSRDUSO - Prepare the dump
 BLSRADDR - Resolve address PDEs
 BLSRADD2 - Generate address space information
 BLSUVSPO - Point to desired record
 BLSUVSGU - Get a record from the dump directory
 BLSUVSEN - Terminate a VSAM request string
 BLSRSAG - Process storage address record
 BLSUVSPO - Point to desired record
 BLSUMPK1 - Compress a message
 BLSUPUTV - Route variable-length message
 BLSUVSEN - Terminate a VSAM request string.
 BLSUFRE1 - Free automatic storage

Figure 94. SCAN Subcommand Control Flow

For lower levels of control flow, refer to the individual module descriptions.

BLSRSADE: DROPMAP Subcommand Processor

Function: BLSRSADE drops entries from the storage map for the currently selected dump. It parses the command, sequentially reads the map within the range specified, and deletes all the records encountered.

Modules Called: BLSRADDR, BLSRADD2, BLSRDUSO, BLSRRDAR, BLSRVPAD, BLSRVPAS, BLSRVPCP, BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUPARI, BLSUPARU, BLSUPGMC, BLSUTRMV, BLSUVP31, BLSUVP32, BLSUVP33, BLSUVSCR, BLSUVSEN, BLSUVSER, BLSUVSGU, BLSUVSPO

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: None.

Data Areas Used: RPL

BLSRSALI: LISTMAP Subcommand Processor

Function: BLSRSALI lists the storage map of the currently selected dump. BLSRSALI then parses the command and sequentially reads and deletes all the entries in the specified range.

Modules Called: BLSRADDR, BLSRADD2, BLSRDATAK, BLSRDUSO, BLSRPADS, BLSRVPAD, BLSRVPAS, BLSRVPCP, BLST01, BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUPARI, BLSUPARU, BLSUPGMC, BLSUPUTV, BLSUVP31, BLSUVP32, BLSUVP33, BLSUVSCR, BLSUVSEN, BLSUVSGU, BLSUVSPO

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: None.

Data Areas Used: RPL

BLSRSCAN: SCAN Subcommand Processor

Function: BLSRSCAN initiates scan probes from map entries within the specified range of addresses. It parses the command, and reads entries sequentially from the map in the range specified. It initiates a scan probe to the specified depth from each entry that is not completely validated. It terminates the scan when the number of new entries in the map has reached the scan limit, or when no new entries are created after a complete pass through the map in the specified range, or the specified number of passes have been made.

Modules Called: BLSRADDR, BLSRADD2, BLSRDUSO, BLSRSAG, BLSRVPAD, BLSRVPAS, BLSRVPCP, BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUPARI, BLSUPARU, BLSUPUTV, BLSUVP21, BLSUVP31, BLSUVP32, BLSUVP33, BLSUVSCR, BLSUVSEN, BLSUVSGU, BLSUVSPO

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: None.

Data Areas Used: RPL

DROPSYM and LISTSYM Subcommands

The module flow through the DROPSYM and LISTSYM subcommands is shown in Figure 95 and Figure 96 on page 235.

BLSRESDR - IPCS DROPSYM main processor BLSUALLO - Allocate automatic storage BLSUPARI - IKJPARS interface BLSRDUSO - Prepare dump for use BLSUVSCR - Create a request parameter list BLSUPGMC - Call non-resident IPCS program BLSRSYRB - Calculate symbol range bounds BLSUVSPO - Point to desired record BLSUVSGU - Get a record from the dump directory BLSRSSSA - Convert alias to standard symbol BLSUVSER - Erase record from the dump directory BLSUVSEN - Terminate a VSAM request string BLSUPGMC - Call non-resident IPCS program BLSRESRU - Issue diagnostics BLSUMPK1 - Compress text of message BLSUTRMV - Transmit a message BLSUFRE1 - Free automatic storage

Figure 95. DROPSYM Subcommand Control Flow

BLSRLSYM - IPCS LISTSYM main processor
 BLSUALLO - Allocate automatic storage
 BLSUPARI - IKJPARS interface
 BLSUPARU - Process routing operands
 BLSUPADS - Process display operands
 BLSRDUSO - Prepare dump for use
 BLSUVSCR - Create a request parameter list
 BLSUPGMC - Call non-resident IPCS program
 BLSRSYRB - Calculate symbol range bounds
 BLSUVSPO - Reposition the VSAM request string
 BLSUVSGU - Get a record from the dump directory
 BLSRSSSA - Change alias to standard symbol
 BLSUPRTT - Print file subtitle initiation/termination
 BLSUTRMA - Transmit variable length message to terminal
 BLSUPUTA - Route variable length message
 BLSRMSGA - Address space name formatter
 BLSRMSGB - Data attribute formatter
 BLSRMSGD - Data type formatter
 BLSRSSSC - Compress standard symbol
 BLSUPUTA - Route variable length message
 BLSUVSEN - Terminate a VSAM request string
 BLSUPGMC - Call non-resident IPCS program
 BLSRESRU - Issue diagnostics
 BLSUFRE1 - Free automatic storage

Figure 96. LISTSYM Subcommand Control Flow

BLSRESDR: DROPSYM Subcommand Processor

Function: BLSRESDR drops selected symbols from the symbol table of the currently selected dump. It parses the command, sequentially reads the symbol table, and drops selected symbols as they are encountered.

Modules Called: BLSRDUSO, BLSRESRU, BLSRSSSA, BLSRSYRB, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUMPK1, BLSUPGMC, BLSUTRMV, BLSUVSCR, BLSUVSEN, BLSUVSER, BLSUVSGU, BLSUVSPO

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: None

Data Areas Used: RPL

BLSRESRU: LISTSYM/DROPSYM Range Usage Diagnostics

Function: This module issues diagnostic messages for ranges that are not marked “USED” and “FOUND.”

Modules Called: BLSRSSSA, BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. List of symbol PDEs.

Output: None.

BLSRLSYM: LISTSYM Subcommand Processor

Function: BLSRLSYM lists selected symbols from the symbol table for the currently selected dump. It parses the command, reads symbols from the symbol table and displays those that were requested.

Modules Called: BLSRDUSO, BLSRESRU, BLSRMSGGA, BLSRMSGGB, BLSRMSGD, BLSRPADS, BLSRSSSA, BLSRSSSC, BLSRSYRB, BLST01, BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUMPK1, BLSUPARI, BLSUPARU, BLSUPGMC, BLSUPRTT, BLSUPUTA, BLSUTRMA, BLSUVSCR, BLSUVSEN, BLSUVSGU, BLSUVSPO

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: None

Data Areas Used: RPL

BLSRSYRB: LISTSYM/DROPSYM Symbol Range Bounds

Function: This module gets LISTSYM/DROPSYM symbol range bounds given a list of symbols/symbol ranges (in the form of IKJPARS PDEs). It returns the minimum of the range origins and the maximum of the range ends. It also initializes the symbol PDEs to “NOT FOUND” and “NOT USED.”

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: None

EQUATE and STACK Subcommand

The program control flow showing the modules involved in processing the EQUATE subcommand is presented in Figure 97 and Figure 98 on page 237.

BLSREQU - IPCS EQUATE main processor
BLSUALLO - Allocate automatic storage
BLSUPARI - IKJPARS interface
BLSRVPAD - Address validity check
BLSUVP31 - Validate for integer 1 through 2^{31}
BLSUVP32 - Validate for integer 0 through $2^{31}-1$
BLSUVP33 - Validate for integer 0 through $2^{24}-1$
BLSRVPCP - Parse validity check exit for CPU ident
BLSRVPAS - Parse validity check for ASID ident PDE
BLSUPARU - Process routing operands
BLSRSSSA - Convert alias to standard symbol
BLSRADDR - Resolve address PDEs
BLSRESAR - Add or replace an equate symbol record
BLSUPRE1 - Free automatic storage

Figure 97. EQUATE Subcommand Control Flow

BLSRESST - IPCS STACK main processor
BLSUALLO - Allocate automatic storage
BLSUPARI - IKJPARS interface
BLSRVPAD - Address validity check
BLSUVP31 - Validate for integer 1 through 2 ³¹ -1
BLSUVP32 - Validate for integer 0 through 2 ³¹ -1
BLSUVP33 - Validate for integer 0 through 2 ²⁴ -1
BLSRVPCP - Parse validity check exit for CPU ident
BLSRVPAS - Parse validity check for ASID ident PDE
BLSUPARU - Establish message routing
BLSRADDR - Resolve address PDEs
BLSUPGME - Call non-resident program
BLSRESSS - Add a stack entry
BLSUTRMV - Diagnose stack full condition
BLSUFRE1 - Free automatic storage

Figure 98. STACK Subcommand Control Flow

For lower levels of control flow (for example, calls to common service routines), refer to the individual module descriptions.

BLSREQU: EQUATE Subcommand Processor

Function: BLSREQU resolves an address expression and associates the result with a given symbol. An EQUATE symbol record is added to the session VSAM cluster. This process is also referred to as adding a symbol to the symbol table. The symbol may be made dropable or nondropable, which refers to whether the PURGE operand is needed on the DROPSYM subcommand in order to delete the symbol.

Modules Called: BLSRADDR, BLSRESAR, BLSRSSSA, BLSRVPAD, BLSRVPAS, BLSRVPCP, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPARU, BLSUVP31, BLSUVP32, BLSUVP33

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: None

BLSRESST: STACK Subcommand Processor

Function: BLSRESST resolves an address expression, determines the last stack entry in use, and associates the result with an entry in the stack. The symbol may be made dropable or nondroppable, which refers to whether the PURGE operand is needed on the DROPSYM subcommand in order to delete the symbol.

Modules Called: BLSRADDR, BLSRESSS, BLSRVPAD, BLSRVPAS, BLSRVPCP, BLSUALLO, BLSUFRE1, BLSUPARI, BLSUPARU, BLSUPGME, BLSUTRMV, BLSUVP31, BLSUVP32, BLSUVP33

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: None

RENUM Subcommand

The program control flow showing the modules involved in processing the RENUM subcommand is presented in Figure 99.

BLSRESRE - IPCS RENUM main processor
BLSUALLO - Allocate automatic storage
BLSUPARI - IKJPARS interface
BLSRDUSO - Prepare dump for use
BLSUPGME - Call non-resident program
BLSRESRN - Renumber the stack
BLSUMPKN - Compress processing summary message
BLSUTRMV - Transmit processing summary message
BLSUFRE1 - Free automatic storage

Figure 99. RENUM Subcommand Control Flow

For lower levels of control flow refer to the individual module descriptions.

BLSRESRE: RENUM Subcommand Processor

Function: BLSRESRE rennumbers all stack entries starting with Z00001 and continuing in increments of one.

Modules Called: BLSRDUSO, BLSRESRN, BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUPARI, BLSUPGME, BLSUTRMV

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: None

BLSRESRN: RENUM Service Routine

Function: BLSRESRN adds a symbol to the stack for a dump.

Modules Called: BLSRESPU, BLSUALLO, BLSUFRE1, BLSUVSCR, BLSUVSEN, BLSUVSER, BLSUVSGU, BLSUVSPO

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Two fullwords

Output

1. Active task variable (see “BLSUZZ2” on page 406) is not used to return data to the caller.
2. Number of stack entries.
3. Number of entries renumbered.

ISPF Dialog Programs

This section contains module descriptions for the ISPF dialog programs included with IPCS.

Logical Screen Task Setup

The program control flow showing the modules involved in establishing IPCS data for an ISPF logical screen task is shown in Figure 100.

BLSG - Establish IPCS/ISPF environment BLSUZZ2C - Create task variable BLSGINI2 - Process dialog request BLSUZZ2D - Delete task variable BLSGINI2 - Process dialog request BLSUALLO - Allocate automatic storage BLSUINI3 - Establish IPCS environment ISPLINK - Provide dialog service BLSUINI8 - Exit from IPCS environment BLSUFRE1 - Free automatic storage BLSUSTAS - ESTAE Exit BLSUZZ2D - Delete task variable
--

Figure 100. ISPF/IPCS Environment Control Flow

For lower levels of control flow refer to the individual module descriptions.

BLSG: Logical Screen Task Setup Dialog

Function: Establish an IPCS environment for an ISPF dialog task.

Modules Called: BLSUZZ2C, BLSGINI2, BLSUZZ2D, BLSUSTAS

Input: The ISPF SELECT service to be performed after setup is complete.

Output: None

BLSGINI2: Logical Screen Task Setup Processor

Function: Establish an IPCS environment for an ISPF dialog task.

Modules Called: BLSUALLO, BLSUFRE1, BLSUINI3, BLSUINI8, ISPLINK

Input: The ISPF SELECT service to be performed after setup is complete.

Output: None

IPCS Subcommand Processing Dialog

The program control flow showing the modules involved in establishing IPCS data for an ISPF logical screen task is shown in Figure 101.

BLSGSCMD - Process IPCS subcommand ISPLINK - Request that ISPF return if a dialog error occurs BLSUSTKT - Create TSO environment for subcommand processing BLSUSCM1 - Process IPCS subcommand BLSUSTKD - Clean up TSO environment created by BLSUSTKT

Figure 101. Subcommand Processing Dialog Control Flow

For lower levels of control flow refer to the individual module descriptions.

BLSGSCMD: IPCS Subcommand Processing Dialog

Function: Process an IPCS subcommand or CLIST for an ISPF logical screen task.

Modules Called: BLSUSCM1, BLSUSTKD, BLSUSTKT, ISPLINK

Input: The command to be executed.

Output: None

Dump Display Reporter - NTRCEPT

The program control flow showing the modules used to alter the dump data in full-screen mode is shown in Figure 102 on page 241.



BLSLMON: Display Screen Data and Process Command

Modules Called: BLSLSTRG, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUVSWB

Input: Active task variable (see “BLSUZZ2” on page 406) points to the NTRCEPT communication block via the ZZ2NTRCP field. The NTRC block contains addresses for storage (obtained via the GETMAIN macro instruction), current cell information, and all common communication fields.

Output: None



BLSLSTRG: Access Data Storage Cells

Function: BLSLSTRG obtains all the main storage that is used to present the dump display reporter lines. BLSLSTRG maintains the available storage using chaining that results in a service very similar to cell pools. BLSLSTRG is called each time a cell is needed to obtain additional storage.

Modules Called: BLSUALLO, BLSUFRE1, ISPLINK

Input: Active task variable (see “BLSUZZ2” on page 406) that contains a pointer (ZZ2NTRCP) to the NTRCEPT communication block

Output: None

Data Areas Used: None

BLSLSNTX: Syntax Check All Line Commands

Function: BLSLSNTX performs syntax checks of the data cells that are passed one at a time from module BLSLMON. Depending on the information in the NTRC block, BLSLSNTX checks for invalid line commands. If any errors are found, ISPF comments are displayed.

Modules Called: BLSUALLO, BLSUFRE1, ISPLINK

Input: Active task variable (see “BLSUZZ2” on page 406) that contains a pointer (ZZ2NTRCP) to the NTRCEPT communication block

Output: None

Data Areas Used: None

BLSLDXCL: Process DELETE and EXCLUDE Line Commands

Function: BSLDXCL performs the following processing: searches through the list and processes each line command; processes the EXCLUDE and DELETE line commands; and finally, creates a message line to replace any lines excluded. BSLDXCL calls BLSLSHOW to process all forms of the SHOW command.

Modules Called: BLSLSHOW, BLSUALLO, BLSUFRE1, BLSUPGME

Input: Active task variable (see “BLSUZZ2” on page 406) that contains a pointer (ZZ2NTRCP) to the NTRCEPT communication block

Output: None

Data Areas Used: None

BLSLSHOW: Process SHOW Primary Command

Function: BLSLSHOW processes all forms of the SHOW line command that redisplay currently excluded lines. If lines are to remain excluded, BLSLSHOW also updates the corresponding message line.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPGME, ISPLINK

Input: Active task variable (see “BLSUZZ2” on page 406) that contains a pointer (ZZ2NTRCP) to the NTRCEPT communication block

Output: None

Data Areas Used: None

BLSLRSET: Process RESET Primary Commands

Function: BLSLRSET clears the entire report, which is generated to this point, of any current line commands and redisplay any excluded lines.

Modules Called: BLSUALLO, BLSUFRE1

Input: Active task variable (see “BLSUZZ2” on page 406) that contains a pointer (ZZ2NTRCP) to the NTRCEPT communication block

Output: None

Data Areas Used: None

BLSLFIND: Process Primary FIND Commands

Function: BLSLFIND performs a syntax check of the primary FIND command. After the command line is parsed and the position of the cursor is calculated, BLSLFIND calls BLSRFISE to search for requested text. BLSLFIND is a secondary entry point that accesses a data buffer.

Modules Called: BLSRVAL, BLSRFISE, BLSUALLO, BLSUFRE1, BLSUPGME, ISPLINK

Input: Active task variable (see “BLSUZZ2” on page 406) that contains a pointer (ZZ2NTRCP) to the NTRCEPT communication block

Output: None

Data Areas Used: None

ISPF Dump Viewing Dialog

The dump viewing dialog makes use of the variable services and display services of ISPF. The control flow for the ISPF dump viewing dialog is shown in Figure 103 on page 245.

BLSLDISP - Display a full screen of dump data
 BLSUPGMU - Load non-resident program
 BLSUSTKT - Stack terminal element
 ISPLINK(VDEFINE) - Define dialog variables to ISPF
 ISPLINK(CONTROL) - Refresh screen and return error
 BLSUZZ2R - Refresh the task variable
 BLSRMDSN - Format dump source
 BLSUPGME - Call non-resident program
 BLSRRDGE - Get RD record
 BLSRMSGA - Format current address space
 BLSUPGME - Call non-resident program
 BLSLMON - Dump display reporter
 BLSUVSWB - Write updated IPCSDDIR buffers
 ISPLINK(DISPLAY) - Display panel BLSPOPT
 ISPLINK(VGET) - Retrieve command variables
 ISPLINK(SELECT) - Perform dialog function
 BLSGSCMD - Process IPCS subcommand
 BLSLIDSN - Determine dump to be browsed
 BLSLIADX - Resolve address expression
 BLSLPTRS - Display pointer stack
 BSLDSTG - Display storage
 ISPEXEC(VDELETE) - Delete ISPF dialog variables

 BSLDSTG - Display storage
 BLSUALLO - Allocate automatic storage
 BLSRMSGA - Format current address space
 ISPLINK(PQUERY) - Determine size of BLSDATA
 ISPLINK(VDEFINE) - Define ISPF variable BLSDATA
 BLSRESGE - Retrieve definition of X
 BLSRRAGE - Locate storage in the dump
 BLSRACCO - Gather dump storage from multiple records
 BLSRACCX - Locate storage in dump buffer
 BLSUPGME - Call non-resident program
 BLSLMON - Dump display reporter
 BLSUVSWB - Write updated IPCSDDIR buffers
 ISPLINK(DISPLAY) - Display dialog entry panel
 ISPLINK(VGET) - Retrieve command variables
 BLSRESAR - Update definition of symbol
 BLSUVSCR - Create a VSAM RPL
 BLSUVSGE - Retrieve CURSOR definition
 BLSUVSER - Delete CURSOR definition
 ISPLINK(SELECT) - Perform dialog function
 BLSGSCMD - Process IPCS subcommand
 BLSRDUCK - Prepare dump for use
 BLSLIADX - Resolve address expression
 BLSRDATK - Edit storage description
 BLSUPGME - Call non-resident program
 BLSRESSS - Add symbol to stack
 BLSUPGME - Call non-resident program
 BLSRESRN - Renumber the stack
 BLSLFISE - Process FIND/RFIND request
 ISPLINK(VDELETE) - Delete ISPF variable BLSDATA
 BLSUFRE1 - Free automatic storage

 BLSLFISE - Process FIND/RFIND request
 BLSUALLO - Allocate automatic storage
 BLSRVAL - Resolve general value
 BLSUPGME - Call non-resident program
 BLSUSIGR - Parse integer subfield
 BLSRFISE - Search dump storage
 BLSUFRE1 - Free automatic storage

Figure 103 (Part 1 of 2). ISPF Dump Viewing Dialog Control Flow

BLSLIADX	- Process address expression
BLSLIADS	- Process address space text
BLSUALLO	- Allocate automatic storage
BLSUPGME	- Call non-resident program
BLSUSIGR	- Parse integer subfield
BLSRMSG	- Format address space
BLSRADDR	- Resolve address expression
BLSUFRE1	- Free automatic storage
BLSLIDSN	- Process dump source
BLSUALLO	- Allocate automatic storage
BLSRDUCK	- Prepare dump for use
BLSRMDSN	- Format dump source
BLSLIADS	- Process default address space
BLSUFRE1	- Free automatic storage
BLSLPTRS	- Display pointer stack
BLSUALLO	- Allocate automatic storage
BLSUVSCR	- Create a VSAM RPL
ISPLINK(PQUERY)	- Determine size of BLSPPTR
ISPLINK(VDEFINE)	- Define ISPF variable BLSPPTR
BLSUVSMR	- Modify a VSAM RPL
BLSUVSPO	- Establish context for next VSAM GET
BLSUVSGU	- Retrieve a VSAM record
BLSRMSG	- Format address space
BLSUVSEN	- Terminate a VSAM request string
BLSUPGME	- Call non-resident program
BLSLMON	- Dump display reporter
BLSUVSWB	- Write updated IPCSDDIR buffers
ISPLINK(DISPLAY)	- Display dialog entry panel
ISPLINK(VGET)	- Retrieve command variables
BLSRESGE	- Verify existence of a stack symbol
BLSLIADX	- Resolve address expression
BLSLIADS	- Resolve address space
BLSUVSER	- Delete stack symbol
BLSRESAR	- Update definition of stack symbol
BLSRDATA	- Edit storage description
BLSRESPU	- Create new stack symbol
ISPLINK(SELECT)	- Perform dialog function
BLSGSCMD	- Process IPCS subcommand
BLSRDUCK	- Prepare dump for use
BLSUPGME	- Call non-resident program
BLSRESSS	- Add symbol to stack
BLSUPGME	- Call non-resident program
BLSRESRN	- Renummer the stack
BLSLDSTG	- Display storage
ISPLINK(VDELETE)	- Delete ISPF variable BLSPPTR
BLSUFRE1	- Free automatic storage

Figure 103 (Part 2 of 2). ISPF Dump Viewing Dialog Control Flow

The current default dump data set and address space are displayed on an entry panel. The user can specify different values and an address. IPCS processing maintains and displays a list of pointers for selection or modification for the purpose of defining subsequent storage displays. Pointers selected from the display can be added to the pointer list. User-entered line and primary commands are processed.

BLSLDISP: Full Screen Dump Viewing Dialog

Function: BLSLDISP displays a full screen of dump data using ISPF services. BLSLDISP displays and processes the input from panel BLSPOPT, the entry panel for the dialog. BLSLDISP calls other modules to parse and handle data entered from panel BLSPOPT (BLSLIDSN) and to display and handle other dialog panels (BLSLPTRS and BLSLDSTG).

Modules Called: BLSGSCMD, BSLDSTG, BSLIADX, BSLIDSN, BSLMON, BSLPTRS, BSRDUCK, BSRMDSN, BSRMSGGA, BSRRDGE, BLSUPGME, BLSUPGMU, BLSUSTKT, BLSUVSWB, BLSUZZ2R, ISPLINK

Input: BLRPRM macro

Output: Display of dump data

BSLDSTG: Display Panel BLSPDISD

Function: BSLDSTG displays and processes the input from panel BLSPDISD, which is the storage panel for the dialog.

Modules Called: BLSGSCMD, BSLFISE, BSLIADX, BSLMON, BSRACCQ, BSRACCX, BSRDATK, BSRDUCK, BSRRESAR, BSRRESGE, BSRRESRN, BSRRESSS, BSRMSGGA, BSRRRAGE, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUVSCR, BLSUVSER, BLSUVSGE, BLSUVSWB, ISPLINK

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Dialog interface block (see “BLSLBLS” on page 346).

Output

1. An updated active task variable (see “BLSUZZ2” on page 406).
2. An updated dialog interface block (see “BLSLBLS” on page 346).

None

BSLFISE: BSLDISP Dialog - FIND/RFIND Processor

Function: BSLFISE parses and processes FIND and RFIND primary commands entered from panel BLSPDISD, which is the storage panel for the dialog.

Modules Called: BLSRFISE, BLSRVAL, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUSIGR

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Dialog interface block (see “BLSLBLS” on page 346).

Output: An appropriate error message identifier in the BLSLBLS, the current dialog location (BLSES) updated to reflect the data found.

BSLIADS: Process Address Space

Function: BSLIADS processes the entry of an address space from a BSLDISP dialog panel.

Modules Called: BLSRADDR, BSRMSGGA, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUSIGR

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Dialog interface block (see “BLSLBLS” on page 346).

Output: An appropriate error message identifier in the BLSLBLS, the address space buffer (BLSAS) that is updated to contain the standard address space description, and the IPCS address space designation (BLSOAS) updated to reflect the designated address space.

BLSLIADX: Process Address Expression

Function: BLSLIADX is a secondary entry point in BLSLIADS. BLSLIADX resolves the address expression.

Modules Called: See BLSLIADS

Input: See BLSLIADS

Output: An appropriate error message identifier in the BLSLBLS, the current dialog location (BLSES) updated to reflect the designated address expression.

BLSLIDSN: Process Panel BLSPOPT Input

Function: BLSLIDSN processes data (the dump source) that is entered on the body of panel BLSPOPT.

Modules Called: BLSLIADS, BLSRDUCK, BLSRMDSN, BLSUALLO, BLSUFRE1

Input

1. Active IPCS task variable (see “BLSUZZ2” on page 406)
2. Dialog interface block (see “BLSLBLS” on page 346).

Output: ZZ2 updated to include the designated DSNAME and when appropriate, an error message identifier.

BLSLPTRS: Display Panel BLSPDISP

Function: BLSLPTRS displays and processes input from panel BLSPDISP, the pointer panel for the dialog.

Modules Called: BLSGSCMD, BLSLDSTG, BLSLIADS, BLSLIADX, BLSLMON, BLSRDATK, BLSRDUCK, BLSRESAR, BLSRESGE, BLSRESPU, BLSRESRN, BLSRESSS, BLSRMSGGA, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUVSCR, BLSUVSER, BLSUVSGU, BLSUVSMR, BLSUVSPO, BLSUVSPU, BLSUVSWB, ISPLINK

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Dialog interface block (see “BLSLBLS” on page 346).

Output

1. An updated active task variable (see “BLSUZZ2” on page 406).
2. An updated dialog interface block (see “BLSLBLS” on page 346).

Common Service Functions

This section contains module descriptions for the IPCS common service functions.

Data Access Services

The program control flow charts for the data access services function are presented in Figure 104 through Figure 106 on page 250 .

- BLSCALOC** - Data access services allocation top level
 - BLSCADST** - Conditional allocate validity check
 - BLSCAINT** - Initialize the DMCB
 - BLSDMSG0** - Message writer
 - BLSCABL** - Build allocation model override parameter list
 - BLSDMSG0** - Message writer
 - BLSCAMER** - Merge model parameters with override parameters
 - BLSDMSG0** - Message writer
 - BLSCAAMS** - Prepare to allocate new VSAM data set
 - IDCAMS** - Allocate VSAM data set
 - BLSCAMIN** - IDCAMS SYSIN interface
 - BLSCAMPR** - IDCAMS SYSPRINT interface
 - BLSDMSG0** - Message writer
 - BLSCHOK** - Hook message to chain
 - BLSCGMF** - Build GETMAIN failure message
 - BLSCHOK** - Hook message to chain
- BLSCADYN** - Connect data set to requestor via SVC99
 - BLSCANAL** - Build dynamic allocation error messages
 - IKJEFF18** - TSO DAIRFAIL message builder
 - BLSDMSG0** - Message writer
 - BLSCRFM** - Reformat TSO messages
 - BLSCHOK** - Hook message to chain
 - BLSCGMF** - Build GETMAIN failure message
 - BLSCHOK** - Hook message to chain

Figure 104. Allocation Control Flow

- BLSCFREE** - Data access services deallocation top level
- BLSRCTCB** - Obtain current IPCS task variable
- BLSCFDYN** - Disconnect data set from requestor via SVC 99
 - BLSCANAL** - Build dynamic deallocation error messages
 - IKJEFF18** - TSO DAIRFAIL message builder
 - BLSDMSG0** - Message writer
 - BLSCRFM** - Reformat TSO messages
 - BLSCHOK** - Hook message to chain
 - BLSCGMF** - Build GETMAIN failure message
 - BLSCHOK** - Hook message to chain
- BLSCFAMS** - Prepare to delete a VSAM data set
 - IDCAMS** - Delete a VSAM data set
 - BLSCAMIN** - IDCAMS SYSIN interface
 - BLSCAMPR** - IDCAMS SYSPRINT interface
 - BLSDMSG0** - Message writer
 - BLSCHOK** - Hook message to chain
 - BLSCGMF** - Build GETMAIN failure message
 - BLSCHOK** - Hook message to chain

Figure 105. Deallocation Control Flow

BLSCRQST - Data access services access request top level
 BLSCRTCB - Obtain current IPCS task variable
 BLSCOPEN - Open data set for access
 BLSCVALT - Data access services open request validation
 BLSDENQ0 - Data set ENQUEUE/WAIT
 BLSDWAIT - Wait on data set in use
 BLSCRECV - OPEN/CLOSE ESTAE recovery
 BLSCCLSE - Close data set to access
 BLSCRECV - OPEN/CLOSE ESTAE recovery
 BLSCSETT - Set VSAM RPL parameters
 BLSCGETT - Get a record
 BLSCSETT - Set VSAM RPL parameters
 BLSCPUTT - Put a record
 BLSCSETT - Set VSAM RPL parameters
 BLSCPOIN - Point to a record
 BLSCSETT - Set VSAM RPL parameters
 BLSCERSE - Erase a record
 BLSCSETT - Set VSAM RPL parameters
 BLSCENDD - Cancel a GET for update record lock
 BLSCSETT - Set VSAM RPL parameters
 BLSCANLE - Build access request error messages
 BLSDMSG0 - Message writer
 IKJEFF19 - TSO VSAMFAIL message builder
 BLSCANLI - VSAMFAIL to TSO message writer intercept
 IKJEFF02 - TSO message writer
 BLSDMSG0 - Message writer
 BLSCRFM - Reformat TSO messages
 BLSCHOK - Hook message to chain
 BLSGMF - Build GETMAIN failure message
 BLSCHOK - Hook message to chain

Figure 106. Data Access Request Control Flow

For lower levels of control flow (for example, calls to common service routines), refer to the individual module descriptions.

BLSCAAMS: VSAM Access Method Services Interface

Function: This module suballocates VSAM data spaces for new VSAM data sets as follows:

1. Obtains the requested data set name from the DMCB.
2. Builds an IDCAMS command stream requesting data set allocation.
3. Builds a group of invocation control blocks for IDCAMS. This includes provision to handle IDCAMS SYSIN and SYSPRINT data using incore buffers by interfacing IDCAMS with BLSCAMIN and BLSCAMPR incore data handlers.
4. Invokes IDCAMS (access method services).
5. Examines the IDCAMS return code:
 - If allocation is successful, updates the DMCB.
 - If allocation fails, formats the IDCAMS incore SYSPRINT data into a chain of messages to be passed back to the requestor.
6. Returns to caller.

Modules Called: IDCAMS, BLSCAMIN, BLSCAMPR, BLSCAMOD, BLSDMSGs, BLSDMSG0, BLSCGMF, BLSCHOK.

Input: DMCB

Output

- For return code 0, none - allocation continues.
- For return codes 4 and 8, the address of the chain of messages issued by IDCAMS is returned to the caller.

BLSCABLD: Build Dynamic Allocation Text Units

Function: This module builds dynamic allocation parameter list text from the allocation parameter list model override entries, and returns addressability for the text unit list to the user in the DMAL control block. It stores selected override parameter values in appropriate DMCB fields to make them available to other functions.

Modules Called: BLSDMSGs, BLSDMSG0

Input: DMCB address in register 1. Also, the allocation model override parameter list, which is an array of fullwords pointed to by the DMALPRMP field.

Output

- For return code 0, the DMALDRBP and DMALDYRB areas of the DMAL control block are initialized to provide addressability to the text unit list by BLSCAMER.
- For return code 8, the DMALDRBP is set to zero.

BLSCADST: Data Access Services Conditional Allocation Dsname Validation

Function: This module ensures that the data set name specified with a conditional allocation request is valid for the request.

Input: A parameter list containing a pointer to the DMCB.

Output: None

BLSCADYN: Dynamic Allocation Interface

Function: This module connects the data set to the requestor as follows:

1. Obtains the address of the dynamic allocation request block from DMALDRBP.
2. Completes the allocation or information retrieval request block setup.
3. Issues the DYNALLOC SVC (SVC99). Examines the DYNALLOC return code:
 - If allocation or information retrieval is successful, it updates the data set related fields of the DMCB.

- If allocation fails, it performs allocation error analysis by calling **BLSCANAL** which returns a chain of messages to be passed back to the requestor.

4. Returns to caller.

Input: DMCB

Output

- For return code 0, the allocation/deallocation section of the DMCB is completed.
- For return code 8, the address of the chain of messages issued by the error analysis module is returned to the caller.

Data Areas Used: IEFZB4D0, IEFZB4D2

BLSCAINT: Data Access Services Allocation Initialization

Function: This module performs data set allocation initialization as follows:

1. Validates the user-supplied allocation model name pointer.
2. Locates the requested allocation model in the BLSCAMOD CSECT and saves its address in the DMALMODP field.
3. Initializes the DMCB and DMAL blocks with the appropriate fields from the model (the DMAL is an extension of the DMCB).
4. Provides default DCB OPEN exit and DCB ABEND exit procedures for non-VSAM data sets.
5. Provides and initializes a DCB exit list for non-VSAM data sets.
6. Returns to caller.

Input: DMCB

Output

- For return code 0, the DMCB fields are initialized and the DMALMODP field is initialized with the address of the requested model.
- For return code 8, the DMALMODP field is set to zero.

BLSCALOC: Data Access Services Data Set Allocation Top Level

Function: This module processes a data set allocation request as follows:

1. For conditional allocation requests, it tests if allocation is required, and if not, returns.
2. It obtains space for all data access functions for this data set.

3. Locates the requested allocation model in BLSCAMOD and initializes the data access services control block (DMCB including the DMAL) by calling the BLSCAINT function.
4. Builds dynamic allocation text units from the allocation override keyword parameter list entries by calling BLSCABLD.
5. Merges the allocation parameter overrides with the selected model allocation parameter list by calling BLSCAMER.
6. If the allocation request is for new data access services space, obtains it by calling the BLSCAAMS function (access method services interface).
7. Connects the data set to the filename by calling the BLSCADYN function (dynamic allocation interface).
8. Hooks the DMCB to the end of the DMCB chain anchored at ZZ1DMCBP.
9. Returns to requestor.

Modules Called: BLSCAAMS, BLSCABLD, BLSCADYN, BLSCAINT, BLSCAMER, BLSDMSG0, BLSDMSG0, BLSUALLO, BLSUFRE1

Input: A parameter list containing:

1. Active task variable (see “BLSUZZ2” on page 406).
2. The address of the DMCB pointer.
3. The allocation model name.
4. The data access mode.
5. The type of allocation request.
6. Allocation model override parameters.

Output

- For return codes 0 and 4, the DMCBP field pointed to by the parameter list contains the address of the data access services control block. BLSCFREE will FREEMAIN the space.

For return code 4, register 0 and DMCBMSG contain the address of a single-thread FIFO message list. This list contains one or messages that describe the reason for the nonzero return code. The requestor is responsible to FREEMAIN the storage obtained for the message chain. BLSDMSG0 may be used to put out the chain and FREEMAIN it.

- For return codes 8 and 12, the DMCBP field contains zero.

For return code 8, register 0 contains the address of a single-thread FIFO message list.

BLSCAMER: Merge Allocation Model Overrides With Model

Function: This module prepares the dynamic allocation parameter list:

1. Obtains addressability to the requested allocation model.
2. Merges the override text units with the model text units giving priority to the override text units.
3. Returns addressability to the completed list to the caller.

Modules Called: BLSCAMOD, BLSDMSGs, BLSDMSG0

Input: DMCB

Output

- For return code 0, the DMALDRBP and DMALDYRB areas of the DMCB are initialized to permit a dynamic allocation request to be issued.
- For return code 8, the DMALDRBP is set to zero.

Data Areas Used: IEFZB4D0, IEFZB4D2

BLSCAMIN: IDCAMS SYSIN Interface Routine

Function: This module provides command stream SYSIN data for IDCAMS as follows:

Request	Function
OPEN	Initializes its pointers in the user data area and returns to IDCAMS.
GET	Unchains the previously accessed record (if any) from the SYSIN data list header in the user data area and freemains its space. • If no record remains in the user data area, it returns to IDCAMS with a return code of 4. • If a record remains in the user data area, it places the address and length of the current record in the IOINFO list, updates the user data area record pointer to the next record address, and returns to IDCAMS with a return code of 0.
CLOSE	Unchains the previously accessed record (if any) from the SYSIN data list header in the user data area, freemains its space and returns to IDCAMS.

Input: Parameter list items passed are:

User data area

Contains a SYSIN data list header and all required BLSCAMIN intercall storage areas. This area is supplied by IDCAMS' caller in the parameter list passed to IDCAMS as the 'SYSIN' dname data area field of the input/output list. The SYSIN command stream for IDCAMS is chained to the SYSIN data list header as a single-threaded FIFO list.

IOFLAGS

A fullword of flags as follows:

Byte 1

Value	Meaning
0	OPEN
4	CLOSE
8	GET

Byte 2

Bit	Meaning
1...	OPEN for input
..1.	IOINFO contains ddname address on OPEN.

IOINFO

- For OPEN or CLOSE:

DNAME: 8-byte field, left-justified (padded with blanks if necessary) containing the ddname. A ddname of 'SYSIN' is expected.

- For GET:

IDCAMS supplies a two-word list. BLSCAMIN places the address of the command stream record in the first word and the length of the record in the second word.

Output: See Input

BLSCAMOD: Data Access Services Allocation Models

Function: This module supplies static parts of the dynamic allocation parameter lists, OPEN/CLOSE lists and DCB or ACB and RPL models for each data set type known to the data access services.

Input: None

Output: None

BLSCAMPR: IDCAMS SYSPRINT Interface Routine

Function: This module collects and processes the SYSPRINT data stream from IDCAMS as follows:

Request	Function
OPEN	Initializes its pointers in the caller data area and returns to IDCAMS.
PUT	Obtains space for the record presented in the IOINFO list, moves the record into the space obtained, chains it to the SYSPRINT data list in the caller data area, and returns with a return code of 0.
CLOSE	Returns to caller.

Input: Parameter list items passed are:

- Caller data area which contains a single-thread FIFO SYSPRINT data list header and all required intercall storage areas. This area is supplied by IDCAMS' caller in the parameter list passed to IDCAMS as the 'SYSPRINT' dname data area of the input/output list.
- IOFLAGS -- A fullword of flags as follows:

Byte 1

Value	Meaning
0	OPEN
4	CLOSE
12	PUT

Byte 2

Bit	Meaning
.1..	OPEN for output
..1.	IOINFO contains ddname address on OPEN.

Bytes 3 and 4

Record type:

Value	Meaning
0	Normal data record to be written.
other	Message serial number converted to binary if IDC message is to be written.

- IOINFO
 - For OPEN or CLOSE:

DNAME: 8-byte field, left-justified (padded with blanks if necessary) containing the ddname. A ddname of 'SYSPRINT' expected.
 - For PUT:

IDCAMS supplies a two-word list. The address of the SYSPRINT data record is in the first word and the length of the record is in the second word.

Output: The SYSPRINT data record chained to the SYSPRINT data list in the user data area.

BLSCANAL: Dynamic Allocation Error Analyzer Interface

Function: This module builds a DAIRFAIL parameter list, links to the DAIRFAIL module, reformats the resulting message buffers into the correct message format, and returns to caller.

Modules Called: BLSCRFM, BLSDMSGs, BLSDMSG0

Input: DMCB

Output: A chain of messages describing the reason for the problem detected by dynamic allocation.

Data Areas Used: IEFZB4D0, IKJEFFDF

BLSCANLE: Data Access Error Analyzer Interface

Function: For VSAM data access errors, this module builds a VSAMFAIL parameter list including the entry point of the BLSCANLI module as VSAMFAIL's message writer, invokes VSAMFAIL and returns to the caller.

For non-VSAM data access errors, it formats the SYNAD message returned by the data access module into IPCS message format, appends the message(s) to the message chain pointed to by DMCBMSG, and returns to caller.

Input: DMCB

Output: A chain of messages describing the reason for the problem detected by data access modules.

BLSCANLI: Data Access Services TSO Message Writer Intercept

Function

1. This module intercepts the TSO message parameter list issued by IKJEFF19 to force IKJEFF02 to place the messages it generates into extract buffers rather than issue one or more putline commands.
2. It invokes IKJEFF02, the TSO message writer.
3. It reformats the message extract buffers into IPCS message format and hooks them to the DMCBMSG chain.

Modules Called: BLSCRFM, BLSDMSGs, BLSDMSG0, IKJEFF02

Input: MTCPPPL - Contains the address of the DMCB for the data set experiencing the error.

Output: Messages chained from DMCBMSG describing the reason for the problem detected by the data access function.

Data Areas Used: IKJEFFMT

BLSCCLSE: Data Access Services Close

Function: BLSCCLSE determines whether a VSAM or QSAM close is to be performed, sets an ESTAE environment in case of a CLOSE ABEND, performs the proper type of close, issues a FREEPOOL if required, dequeues the data set if required, analyzes the result of the request, and returns the proper return code and the modified DMCB to BLSCRQST. If a VSAM close is requested, a VSAM end request is issued prior to issuing the CLOSE macro.

Input: DMCB

Output: Certain indicators in the DMCB are modified to indicate the result of the request.

BLSCENDD: Data Access Services End Request Executor

Function: BLSCENDD issues an end request against the data set associated with the DMCB, and computes the proper return code.

Input: DMCB

Output: Modified DMCB

BLSCERSE: Data Access Services Erase Executor

Function: BLSCERSE issues an ERASE request against the data set associated with the DMCB, and computes the proper return code.

Input: DMCB

Output: Modified DMCB

BLSCFAMS: Data Access Services Free VSAM Access Method Services Interface

Function: This module frees VSAM data spaces for VSAM data sets with delete disposition as follows:

1. It obtains the requested data set name from the DMCB.
2. It builds an IDCAMS command stream requesting data set deletion.
3. It builds a group of invocation control blocks for IDCAMS. This includes provision to handle IDCAMS SYSIN and SYSPRINT data using incore buffers by interfacing IDCAMS with BLSCAMIN and BLSCAMPR incore data handlers.
4. It invokes IDCAMS (access method services).
5. It examines the IDCAMS return code:
 - If deletion is successful, updates the DMCB.
 - If deletion fails, formats the IDCAMS incore SYSPRINT data into a chain of messages to be passed back to the requestor.
6. Returns to caller.

Input: A parameter list containing:

1. The address of the DMCB.
2. The address of the FREE keyword parameter list.

Output

- For return code 0, none - free continues.
- For return codes 4 and 8, the address of the chain of messages issued by IDCAMS is returned to the caller.

BLSCFDYN: Dynamic Deallocation Interface

Function: This module disconnects the data set from the ddname as follows:

1. It builds a deallocation request block.
2. If a DISP override is supplied, it requests data set release rather than turns off the in-use bit for the data set.
3. It issues the DYNALLOC SVC (SVC 99).
4. It examines the DYNALLOC return code:
 - If deallocation is successful, updates the allocation-dependent fields of the DMCB.
 - If allocation fails, performs deallocation error analysis by calling BLSCANAL which returns a chain of messages to be passed back to the requestor.
5. Returns to caller.

Input: A parameter list containing:

1. Address of the DMCB
2. Address of the FREE keyword parameter list

Output

- For return code 0, the allocation/deallocation section of the DMCB is updated.
- For return codes 4 and 8, the address of the chain of messages issued by the error analysis module is returned to the caller.

Data Areas Used: IEFZB4D0, IEFZB4D2

BLSCFREE: Free (Deallocate) a Data Set

Function: This module performs the data set free function as follows:

1. It disconnects the file name from the data set name by calling BLSCFDYN.
2. It frees data access services space by calling the BLSCFAMS function (VSAM access method services interface).
3. It unhooks the DMCB from the chain anchored at ZZ1DMCBP.
4. If the return code is less than 8, it releases the DMCB and sets the DMCB pointer in the caller's program to zero.
5. It returns to caller.

Input: A 3-entry parameter list containing:

1. DMCB - The address of the DMCB.
2. DMCBP - The name of a pointer variable that contains the address of the DMCB. This parameter will be zeroed if the free is successful.
3. IOPTLIST - The name of a variable-length field that contains the free function override keyword parameter entries. If there are no overrides, the value of the name field is zero.

Output

- For return code 0, the DMCB is released and the DMCBP field is set to zero.
- For return code 4, the DMCB is released and the DMCBP field is set to zero. Registers 0 and 1 provide feedback information:

Reg	Value
1	The dynamic deallocation information reason code that corresponds to the warning message returned in the message list.
0	The address of a message list in the correct problem/data management format, as described in the description of BLSDMSG0.

- For return code 8, the DMCB is not released. Register 0 and DMCBMSG contain the address of a message list which contains one or messages that describe the reason for the nonzero return code. The requestor is responsible to release the storage obtained for the message chain. BLSDMSG0 may be used to output the chain and do the release.

BLSCGETT: Data Access Services Get

Function: BLSCGETT determines the proper type of get request to issue from the type of data set associated with the DMCB and the request modifier byte in the DMCB. BLSCGETT issues the request, and computes the proper return code.

Input: DMCB.

Output: Modified DMCB.

BLSCGMF: Build GETMAIN Fail Message BLS03109

Function: This module builds GETMAIN fail message BLS03109 and hooks it to the requestor's message anchor.

Input

1. DMCB.
2. Address of first message on existing chain or zero.
3. Name of module detecting GETMAIN failure.

Output: Message BLS03109 hooked to MSGANCHR, requestor's message anchor. The requestor is responsible to freemain the storage obtained for the message. The output message list format is described in the BLSDMSG0 module description.

BLSCHOK: Append Message to Message Chain Anchor

Function: This module appends a requestor supplied message to the requestor's message chain anchor.

Input: A parameter list containing:

1. DMCB.
2. Address of first message on existing chain or zero.
3. Address of message to be appended.

Output: Requestor's message appended to anchor's message chain. The requestor is responsible to freemain the storage obtained for the message. The output message list format is described in the BLSDMSG0 module description.

BLSCOPEN: Data Access Open

Function: BLSCOPEN determines whether a VSAM or QSAM open is to be performed, enqueues the data set if required, sets an ESTAE environment in case of OPEN ABEND, performs the proper type of open, analyzes the result of the request, activates a DCB ABEND exit for non-VSAM data sets, and returns the proper return code and the modified DMCB to BLSCRQST. If an OPEN error occurs for a high-contention VSAM data set, BLSCOPEN dequeues the data set.

Input: DMCB

Output: Certain indicators in the DMCB are modified to indicate the result of the request.

BLSCPOIN: Data Access Services POINT Executor

Function: BLSCPOIN issues the proper form of POINT request for the type of data set associated with the DMCB. The result of the point request is analyzed to compute the proper return code.

Input: DMCB.

Output: Modified DMCB.

BLSCPUTT: Data Access Services PUT Executor

Function: BLSCPUTT determines the proper type of PUT request to issue from the type of data set associated with the DMCB and the request modifier byte in the DMCB. BLSCPUTT issues the request, and computes the proper return code.

Input: DMCB.

Output: Modified DMCB.

BLSCRECV: Data Access Services OPEN/CLOSE ABEND Recovery

Function: BLSCRECV provides recovery from ABENDs which may occur during execution of data access services.

Input: SDWA and DMCB

Output: The DMCBCPC field contains the ABEND completion code.

Data Areas Used: SDWA

BLSCRFM: Reformat TSO Message to IPCS Message

Function: This module reformats a TSO message buffer into IPCS message format and hooks it to the requestor's message anchor.

Input

1. DMCB
2. Address of first message on existing chain or zero.
3. Address of TSO message buffer to be reformatted.

Output: Requestor's message reformatted in IPCS message format and appended to ANCH message chain. The requestor is responsible to release the storage obtained for the message. The output message list format is described with BLSMSG0.

BLSCRQST: Data Access Request Interface

Function: This module validates the data access request, determines its nature, and invokes other data access services modules in the proper sequence to satisfy the request. BLSCRQST then returns an indication of the result of the request.

The possible requests are: OPEN, CLOSE, SET, GET, POINT, PUT, ERASE, and END.

Input: DMCB

Output: Certain indicators in the DMCB are modified to indicate the result of the request. In addition, as the result of a successful input request a user-supplied buffer, located by the DMCB, contains the input data.

Return Codes: Register 15 return codes returned to caller:

Code	Explanation
00	Successful completion - all requests
04	Share failure on OPEN request. Permanent error on CLOSE request
08	Recoverable error on OPEN request. End of file or record not found on GET or POINT request. Duplicate key on PUT request. Record not held for update on ERASE request.
12	Permanent error on all requests except CLOSE.

BLSCRTCB: Get ZZ2 Pointer for Current TCB

Function: This module obtains the BLSUZZ1 pointer from DMCBZZ1P. It runs the BLSUZZ2 chain that is anchored on ZZ1ZZ2P until ZZ2TCBP matches the current TCB address. It stores this BLSUZZ2 pointer in DMCBTVP.

Input: DMCB.

Output: DMCBTVP contains the BLSUZZ2 block address for the current TCB.

Data Areas Used: PSA

BLSCSETT: Data Access Services Control Block Modification

Function: BLSCSETT examines a data access services control block for changes made by the requester. Any such changes are checked for validity and consistency. If any changes require the modification of the VSAM control block within the DMCB then BLSCSETT issues the appropriate VSAM MODCB request.

Input: DMCB

Output: Modified DMCB

Data Areas Used: CVT

BLSCVALT: Data Access Services Open Request Validation

Function: This module ensures that the requester is not enqueued upon the data set directory when he requests that the problem data set be opened and enqueued upon.

Input: DMCB.

Output: None.

BLSDADSD: Allocate the Data Set Directory

Function: This module obtains the data set directory data set name from the facility parameters block and issues a data access services conditional allocate request for the data set directory data set. If allocation is successful, it places the DMCB pointer returned by allocation in ZZ1DSDP.

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: ZZ1DSDP is initialized with DMCB pointer value.

BLSDAPDR: Allocate the Problem Directory

Function: This module obtains the problem directory data set name from the facility parameters block and issues a data access services conditional allocate request for the problem directory data set. If allocation is successful, it places the DMCB pointer returned by allocation in ZZ1PDRP.

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: ZZ1PDRP is initialized with DMCB pointer value.

BLSDFDSD: Free the Data Set Directory

Function

1. This module validates the BLSUZZ1 control block.
2. It obtains the address of the data set directory DMCB from ZZ1DSDP. If it is zero or null, indicating that the data set directory is not allocated, it returns to the caller.

3. If ZZ1DSDP is not null, it tests that it points to a valid DMCB. If it does not, issue messages and abend.
4. If the data set directory is open, and this module is not executing under the same TCB that opened the data set directory, it returns to caller.
5. If the data set directory is open, it issues a BLSCLOSE request to close it.
6. If the data set directory is closed, it issues a BLSFREE request to deallocate the data set directory and destroy the DMCB.
7. If the unallocation is successful, ZZ1DSDP is set to null to indicate that the data set directory is deallocated. Any messages generated by data access services are routed to the user prior to return.

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: ZZ1DSDP is set to null if the free is successful.

Data Areas Used: PSA

BLSDFPDR: Free Data Access Services Controlled Data Sets

Function: This module validates the BLSUZZ1 control block. It obtains the address of the DMCB chain from ZZ1DMCBP. If it is null, indicating there are no data sets allocated, it returns to the caller. If ZZ1DMCBP is not null, it points to the DMCB of the first data set to be freed. For each entry on the DMCB chain:

1. It tests that the entry is a valid DMCB. If it is not, it issues messages and abends.
2. If the data set is open, and this module is not executing under the same TCB that opened the data set, it leaves it on the DMCB chain, and goes to the next entry on the chain.
3. If the data set is open, it calls BLSCLOSE to close it.
4. If the data set is closed, it calls BLSCFREE to deallocate it, unchains and destroys the DMCB, and unhooks it from the DMCB chain.

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: When the data set directory is deallocated successfully, field ZZ1DSDP is set to null. When the problem directory is deallocated successfully, field ZZ1PDRP is set to null.

Data Areas Used: PSA

BLSDWAIT: Data Access Services Interruptible Wait Function

Function: This module tests for a pending attention interrupt. If no interrupt, it issues an STIMER wait.

Input: Active task variable (see “BLSUZZ2” on page 406).

Output: None

Print and Terminal Services

The print and terminal services routines route and transmit messages to the active IPCS destination (printer and/or terminal). There are three groups of these routines:

- Print routines
- Message routing routines
- Terminal transmission routines

The discussion of each group of modules follows.

Print Routines

The print routines transmit messages to the printer. They are summarized in Figure 107.

SUPPORTS			

MODULE	VAR LTH	MULTI LINE	INVOKES MODULE
BLSUPRTA	X		BLSUPRTN
BLSUPRTN		X	BLSUPRTA
BLSUPRTT	Accepts a message in multi-line format to be used as the print page sub-title.		BLSUPRTN

Figure 107. Print Transmission Routine Summary

For more detail see the individual module descriptions.

BLSUPRCC: Close IPCSPRNT

Function: Set up environment to issue an inter-task service request. Issue the request to close the print file and wait for the completion of this service. The IPCS print file is closed but not freed.

Modules Called: BLSUALLO, BLSUFRE1

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSUPRTA: Transmit Variable-Length (ASA) Message to Printer

Function: BLSUPRTA transmits (via QSAM - RECFM=VBA) a variable-length message to the print file. Pagination control is provided as follows:

- A new page is initiated as a result of:
 - An explicit page eject request (ASA CC='1').
 - A message which would result in a page overflow condition. The control character associated with this message is replaced with a single-space control character.
- Each new page begins with:
 - A title line followed by one blank line. The text of the title can be supplied in one of these ways:
 1. The IPCS user can use the OPEN subcommand to supply a specific title.
 2. If the user does not supply a title, IPCS attempts to extract the title from the dump data set currently in use and uses that text.
 3. If neither of the preceding sources of title text is available, IPCS uses the text "IPCS PRINT LOG (FOR userid)."
 4. If the userid is not available, IPCS uses the text "IPCS PRINT LOG."
 - If a subtitle is established, IPCS permits one or more subtitle lines. BLSUPRTN is called to transmit the subtitle.

Support for control characters is restricted to page eject, overprint, and single, double, and triple-space; all others are replaced with a single space.

Modules Called: BLSDDTIME, BLSRPHGE, BLSUALLO, BLSUFRE1, BLSUPGMC, BLSUPGME, BLSUPRTA (recursive), BLSUPRTN

Input

1. Active task variable (see "BLSUZZ2" on page 406)
2. A buffer containing the message to be routed

Output: None

Data Areas Used: PSCB, DCB

BLSUPRTN: Transmit Multiline (ASA,MSGID) Message to Printer

Function: BLSUPRTN transmits a multiline message to the printer. The transmission is accomplished one line at a time. The first character of each line of message text is interpreted as an ASA control character.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPRTA

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A buffer containing the message to be routed

Output: None

BLSUPRTT: Print File Subtitle Initiation/Termination

Function: BLSUPRTT initiates/terminates the definition of a print file subtitle. The subtitle is specified in the form of a TSO multiline message (each message line is prefixed by an ASA control character). Subtitle termination is signaled by a null message (length<5). A subtitle is routed to the print file upon initiation and on each subsequent occurrence of a new print file page, until the definition of the subtitle is terminated.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPRTN

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A buffer containing the message to be routed

Output: None

Notes:

1. IKJPUTL is a TSO service function.
2. Prior to invoking BLSUPRTA, BLSUPUTD and BLSUPUTV prefix a single-space control character to the message.

Message Routing Routines

The message routing routines transmit messages to the printer, the terminal, or both. They are summarized in Figure 108 on page 268 .

MODULE	SUPPORTS				ROUTING	
	CTRL CHAR	MSG ID	VAR LTH	MULTI LINE	DEST	INVOKES MODULE
BLSUPUTA	X		X		P T	BLSUPRTA BLSUTRMA
BLSUPUTC	X			X	P,T	IKJPUTL(1) BLSUPUTA
BLSUPUTD(2)			X		P T	BLSUPRTA IKJPUTL(1)
BLSUPUTI			X		P,T	BLSUPUTT
BLSUPUTN	X	X			P T	BLSUPRTN BLSUTRMN
BLSUPUTO		X		X	P,T	IKJPUTL(1) BLSUPUTV
BLSUPUTT			X		P,T	BLSUPUTA
BLSUPUTV(2)	X	X	X		P T	BLSUPRTA BLSUTRMV

P = printer, T = terminal

Figure 108. Message Routing Routine Summary

For more detail see the individual module descriptions.

BLSUPUTA: Route Variable-length (ASA) Message

Function: BLSUPUTA routes a variable-length message to the terminal, the print file, or both, as indicated by the settings of flags ZZ2AFP and ZZ2AFT. The message is in the format required by the IKJPTGT and IKJPUTL TSO service routines. The first character of the message text is interpreted as an ASA control character.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPRTA, BLSUTRMA

Input

1. Active task variable (see "BLSUZZ2" on page 406)
2. A buffer containing the message to be routed

Output: None

BLSUPUTC: Route Single-level (ASA) Message

Function: BLSUPUTC routes a single-level message to the terminal, the print file, or both, as indicated by the settings of flags ZZ2AFP and ZZ2AFT. The message is specified using an output line descriptor formatted as required by the IKJPTGT and IKJPUTL TSO service routines. An ASA control character appears as the first character of the message text.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPUTA, IKJPUTL

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. An output line descriptor (OLD) for the message to be routed.

Output: None

Data Areas Used: ECT, UPT

BLSUPUTD: Route Variable-length (Data Only) Message

Function: BLSUPUTD routes a variable-length message to the terminal, the print file, or both, as indicated by the settings of flags ZZ2AFP and ZZ2AFT. The message is in the format required by the IKJPTGT and IKJPUTL TSO service routines and consists solely of the data to be displayed. No message identifier or ASA control character is present. A single-space ASA control character is prefixed to the message prior to transmission to the printer.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPUTV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A buffer containing the message to be routed

Output: None

Data Areas Used: ECT, UPT

BLSUPUTI: Message Router

Function: Message router interface for MVS/XA.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPUTT

Input: Control may be received in AMODE(24) or AMODE(31). BLSUPUTI is entered only when IPCS is active in an MVS/XA host environment.

Reg	Value
0	Multiple uses depending upon the type of storage requested.
1	Active task variable (see “BLSUZZ2” on page 406)
13	Address of a 72-byte save area which may be above 2 ²⁴ -1 when BLSUPUTI is entered in AMODE(31).
14	Return address
15	Address of BLSUPUTI

Output: Address of the data in the buffer is returned in register 0.

BLSUPUTN: Route Multiline (ASA, MSGID) Message

Function: BLSUPUTN routes a multiline message to the terminal, the print file, or both, as indicated by the settings of flags ZZ2AFP and ZZ2AFT. The message is in the format required by the IKJPTGT and IKJPUTL TSO service routines. The first character of the message text is interpreted as an ASA control character. Message ID stripping is supported.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPRTN, BLSUTRMN

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A buffer containing the message to be routed.

Output: None

BLSUPUTO: Route Single-level (MSGID) Message

Function: BLSUPUTO routes a single-level message to the terminal, the print file, or both, as indicated by the settings of flags ZZ2AFP and ZZ2AFT. The message is specified using an output line descriptor formatted as required by the IKJPTGT and IKJPUTL TSO service routines. Message ID stripping is supported.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPUTV, IKJPUTL

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A TSO output line descriptor (OLD) for a single-level message

Output: None

Data Areas Used: ECT, UPT

BLSUPUTT: Simulate PRDMP Print Service Routine

Function: BLSUPUTT simulates the function of the PRDMP print service routine by routing the contents of the output buffer to the printer, the terminal, or both, as indicated by the settings of flags ZZ2AFP and ZZ2AFT.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPUTA, BLSUTRMA

Input: Control must be received in AMODE(24). BLSUPUTT is entered directly when IPCS is active in an MVS/370 host environment, indirectly from BLSUPUTI when IPCS is active in an MVS/XA host environment.

Reg	Value
0	Multiple uses depending upon the type of storage requested.
1	Active task variable (see “BLSUZZ2” on page 406)
13	Address of a 72-byte save area which must be below 2 ²⁴ .
14	Return address
15	Address of BLSUPUTT

Output: The output buffer is blanked.

BLSUPUTV: Route Variable-Length (MSGID) Message

Function: BLSUPUTV routes a variable-length message to the terminal, the print file, or both, as indicated by the settings of flags ZZ2AFP and ZZ2AFT. The message is in the format required by the IKJPTGT and IKJPUTL TSO service routines. Message ID stripping is supported. A single-space control character is prefixed to the message prior to transmission to the printer.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPRTA, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A buffer containing the message to be routed

Output: None

Notes:

1. BLSUTRMA and BLSUTRMN honor spacing control characters by generating the necessary number of blank lines. Control characters are stripped from the message prior to the invocation of BLSUTRMV.
2. IKJPUTL is a TSO service function.

Terminal Transmission Routines

The terminal transmission routines transmit messages to the terminal. They are summarized in Figure 109.

MODULE	SUPPORTS				INVOKES MODULE
	***** CTRL CHAR	***** MSG ID	***** VAR LTH	***** MULTI LINE	
BLSUTRMA(1)	X		X		BLSUTRMV
BLSUTRMN(1)	X	X		X	BLSUTRMV
BLSUTRMV		X	X		IKJPUTL(2)

Figure 109. Terminal Transmission Routine Summary

For more detail see the individual module descriptions.

BLSUTRMA: Transmit Variable-Length (ASA) Message to Terminal

Function: BLSUTRMA transmits a variable-length message to the terminal. The message is in the format required by the IKJPTGT and IKJPUTL TSO service routines. The first character of the message text is interpreted as an ASA control character. Double and triple-space characters are supported by transmitting the appropriate number of blank lines.

Modules Called: BLSUALLO, BLSUFRE1, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A buffer containing the message to be routed

Output: None

BLSUTRMN: Transmit Multiline (ASA, MSGID) Message

Function: BLSUTRMN transmits a multiline message to the terminal. The message is in the format required by the IKJPTGT and IKJPUTL TSO service routines. The first character of each line of message text is interpreted as an ASA control character, which is stripped from the text prior to transmission to the terminal. Double and triple-space characters are supported by transmitting the appropriate number of blank lines.

Modules Called: BLSUALLO, BLSUFRE1, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A buffer containing the message to be transmitted

Output: None

BLSUTRMV: Transmit Variable-Length (MSGID) Message to Terminal

Function: BLSUTRMV transmits a variable-length message to the terminal. The message is in the format required by the IKJPTGT and IKJPUTL TSO service routines. If the initial character(s) of text are non-blank, they form a message identifier. If the initial character is blank, no message identifier is present.

BLSUTRMV and BLSUMON are the two controlling modules in the dump display reporter. When the IPCS dialogs (BLSGSCMD or BLSLDISP) are active in an ISPF environment, BLSUTRMV builds an in-storage list of the messages, which are to be directed to the terminal for display. When enough messages have accumulated to fill the screen, BLSLMON is called to present that portion of the current report. This is done repeatedly for long reports.

Modules Called: BLSLMON, BLSLSTRG, BLSUALLO, BLSUFRE1, BLSUPGME, IKJPUTL

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A buffer containing the message to be transmitted

Output: None

Message Text Build and Routing

The message text build and routing function consists of four modules.

BLSDMSGs: Data Access Message Text CSECT

Function: BLSDMSGs is a non-executable module of message text containing all IPCS problem management, data management, and data access services messages.

Input: None

Output: None

BLSDMSG0: Problem Management Message Text Build

Function: BLSDMSG0 constructs a message from a message skeleton and possibly a set of inserts, and either returns the message to the caller, or routes it to one of three message output routines, and frees it before returning to the caller.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Message number of the skeleton message to be found in the input message CSECT.
3. A fullword containing the address of the first message on the input message chain, or address of a fullword containing zeros if there is no input message chain.
4. The input message CSECT which contains the message skeleton identified by the second input parameter.
5. A flag byte, which contains flag settings to control LIFO/FIFO queueing of the output message on the message chain (bit 0), to compress trailing blanks from inserts (bit 1), and to control routing (bits 2 and 3). Bits 4-7 are reserved.
6. A fullword which contains the address of the first insert element on the input insert chain, or the address of a fullword which contains zeros if there is no input insert chain.

Output: Register 1 points to a 6-word parameter list as described in the input parameters above, but the third parameter contains the address of the first message on the output message chain, if any. It is reset if the caller requested LIFO queueing of the output message. It is set to zero if the caller requested either message routing (flag bits 2 and 3) or requested message chain flush (input message number was minus one).

If the user requested routing (flag bits 2 and 3), the output message chain is put out through one of the three output message routines.

Storage Map Management

The program control flow showing the modules used by the storage map management function is presented in Figure 110 on page 274.

BLSRSAG - Process a storage address record
 BLSUALLO - Allocate automatic storage
 BLSRSAGN - Normalize input storage address record
 BLSRSAGQ - Get active storage address record
 BLSRSAGC - Retrieve storage address record from the dump directory
 BLSUALLO - Allocate automatic storage
 BLSUVSCR - Create a VSAM RPL
 BLSUVSGE - Retrieve the designated dump directory record
 BLSUFRE1 - Release automatic storage
 BLSUPGMC - Call non-resident entry point
 BLSVxxxx - Scan the block identified
 BLSRSAPU - Add storage address record to the dump directory
 BLSUALLO - Allocate automatic storage
 BLSUVSCR - Create a VSAM RPL
 BLSUVSPU - Add the designated dump directory record
 BLSUFRE1 - Release automatic storage
 BLSRSAAR - Add a storage address record to the dump directory if none exists. Otherwise, replace the existing storage address record.
 BLSUALLO - Allocate automatic storage
 BLSUVSCR - Create a VSAM RPL
 BLSUVSAR - Add or replace the designated dump directory record
 BLSUFRE1 - Release automatic storage
 BLSUFRE1 - Release automatic storage
 BLSRSAGE - Retrieve existing storage address record
 BLSUALLO - Allocate automatic storage
 BLSRSAGQ - Get active storage address record
 BLSRSAGC - Retrieve storage address record from the dump directory (see BLSRSAGC flow above)
 BLSUFRE1 - Release automatic storage
 BLSRSAGU - Get storage address record unconditionally
 BLSUALLO - Allocate automatic storage
 BLSRSAG - Process storage address record
 BLSUPGMC - Call non-resident entry point
 BLSVxxxx - Rescan the block identified
 BLSUFRE1 - Release automatic storage
 BLSRSAPC - Identify damaged storage area
 BLSUALLO - Allocate automatic storage
 BLSRMSGD - Format data type text
 BLSRMSGA - Format address space text
 BLSUPUTO - Route message BLS18058I
 BLSRMSGD - Format data type text
 BLSRMSGA - Format address space text
 BLSUPUTO - Route message BLS18059I
 BLSUFRE1 - Release automatic storage
 BLSRESSA - Determine validity of data
 BLSRSAGU - Get storage address record unconditionally
 BLSRESSR - Scan block
 BLSRSAGR - Get SA record
 BLSRSAG - Scan data
 BLSUPGMC - Call non-resident program

Figure 110. Storage Map Management Control Flow

BLSRMVSA: Transmit Unique Scan Messages

Function: Transmit unique messages for scan routines.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPUTV

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. Unique message descriptor
3. Error flag array

Output: None

BLSRSAAR: Add or Replace a Storage Address Record

Function: This module stores a record in the storage map. It replaces an existing record if necessary.

Modules Called: BLSUALLO, BLSUFRE1, BLSUVSCR, BLSUVSAR

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. A storage address record (see “BLSRSASY” on page 391), completely initialized except for field SASYRDX.

Output: Field SASYRDX is returned containing the current dump reference number.

BLSRSAG: Process Storage Address Record

Function: This module obtains a storage address record (see “BLSRSASY” on page 391) pertaining to one area, module, or structure. It obtains this information:

1. From an active record queue, representing data which is actively being scanned.
2. From an existing storage address record in the dump directory.
3. From a routine which has been written to generate a storage address record (scan the data) for the specified type of data.

If no record can be obtained from the preceding sources, it produces one which indicates that the block has default storage characteristics and that no fault was detected in it. It ensures that the map is updated to reflect any new information produced.

Modules Called: BLSRSAGN, BLSRSAGP, BLSUALLO, BLSUFRE1.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. A buffer for a storage address record (see “BLSRSASY” on page 391). If the third parameter is not an equate symbol record, fields SASYAS, SASYLAD, and SASYDT must be initialized. If the third parameter is an equate symbol record, the three fields are obtained from the equate symbol record.
3. One of three items may be supplied:
 - A storage address record (see “BLSRSASY” on page 391) describing a block which addresses the block for which information is to be obtained.

- An equate symbol record (see “BLSRESSY” on page 373) in which fields ESSYAS, ESSYLAD, and ESSYDT have been initialized to locate the block.
- Two or more bytes of binary zeros, indicating that neither of the preceding has been supplied.

Output

1. The requested record is returned.
2. If an equate symbol record was passed, field ESSYD is returned.

BLSRSAGC: Get Storage Address Record

Function: First a request parameter list is created, and used to get the requested record. If the get is successful, a flag is set. Otherwise, the return code is set indicating an error condition has occurred.

Modules Called: BLSUALLO, BLSUFRE1, BLSUVSGE, BLSUVSCR

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Storage address record (see “BLSRSASY” on page 391) model with all fields in the key portion completed.

Output: The storage address record requested.

Data Areas Used: RPL

BLSRSAGE: Get Storage Address Record

Function: An effort is made to retrieve the active storage address record. If that fails, the storage address record is retrieved from the cluster.

Modules Called: BLSRSAGC, BLSRSAGQ, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Storage address record (see “BLSRSASY” on page 391) model with all fields in the key portion completed.

Output: The storage address record requested.

BLSRSAGN: Get Normalized Storage Address Record

Function: This module produces a storage address record which has the properties:

1. MVS/XA common virtual storage is described as ASID(1).
2. One doubleword of scalar storage is described unless that length would cause an unreasonable description to be produced. In that case a smaller length is provided.

3. If the data type is one of a group of related data types the specific data type will appear in the storage description but a common group name will appear in the key of the record. For example, structure UCBDA is changed to structure UCB in the key of the record.
4. If validation of the data type is supported, the name of the program (often described as a scan exit) which performs validation is supplied.

Input: A two-entry parameter list addressing:

1. Active task variable (see “BLSUZZ2” on page 406).
2. Storage address record (see “BLSRSASY” on page 391)

Output: Storage address fields SASYAS, SASYDTD, SASYF, and SASYPGV are updated as required.

BLSRSAGP: Process SA Record

Function: Get SA record.

Modules Called: BLSRDATAK, BLSRSAAR, BLSRSAGC, BLSRSAGN, BLSRSAGQ, BLSRSAPU, BLSUALLO, BLSRFRE1, BLSUPGMC

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. A buffer for a storage address record (see “BLSRSASY” on page 391). Fields SASYAS, SASYLAD, SASYDT, and SASYF must be initialized.
3. One of three items may be supplied:
 - A storage address record (see “BLSRSASY” on page 391) describing a block which addresses the block for which information is to be obtained.
 - An equate symbol record (see “BLSRESSY” on page 373) in which fields ESSYAS, ESSYLAD, and ESSYDT have been initialized to locate the block.
 - Two or more bytes of binary zeros, indicating that neither of the preceding has been supplied.

Output: The second parameter is updated to reflect scan processing performed on the data.

BLSRSAGQ: Get Active Storage Address Record

Function: A search is made of all the active SA records looking for the requested record. If the record is found, it is returned. Otherwise, the return code is set.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Storage address record (see “BLSRSASY” on page 391) model with all fields in the key portion completed.

Output: The storage address record requested.

BLSRSAGR: Get SA Record

Function: After setting the scan depth limit, a call is made to BLSRSAGS to scan for the record.

Modules Called: BLSRSAGS, BLSUALLO, BLSUFRE1, BLSUPGMC

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. A buffer for a storage address record (see “BLSRSASY” on page 391). Fields SASYAS, SASYLAD, SASYDT, and SASYF must be initialized.
3. One of two items may be supplied:
 - A storage address record (see “BLSRSASY” on page 391) describing a block which addresses the block for which information is to be obtained.
 - Two or more bytes of binary zeros, indicating that the preceding has not been supplied.

Output: The second parameter is updated to reflect scan processing performed on the data.

BLSRSAGS: Process SA Record

Function: Get SA record.

Modules Called: BLSRSAGN, BLSRSAGP, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. A buffer for a storage address record (see “BLSRSASY” on page 391). Fields SASYAS, SASYLAD, SASYDT, and SASYF must be initialized.
3. One of two items may be supplied:
 - A storage address record (see “BLSRSASY” on page 391) describing a block which addresses the block for which information is to be obtained.
 - Two or more bytes of binary zeros, indicating that the preceding has not been supplied.

Output: The second parameter is updated to reflect scan processing performed on the data.

BLSRSAGU: Get Storage Address Record Unconditionally

Function: After setting the scan depth limit, a call is made to BLSRSAG to scan for the record.

Modules Called: BLSRSAG,BLSUALLO, BLSUFRE1, BLSUPGMC

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. A buffer for a storage address record (see “BLSRSASY” on page 391). If the third parameter is not an equate symbol record, fields SASYAS, SASYLAD, and SASYDT must be initialized. If the third parameter is an equate symbol record, the three fields are obtained from the equate symbol record.
3. One of three items may be supplied:
 - A storage address record (see “BLSRSASY” on page 391) describing a block which addresses the block for which information is to be obtained.
 - An equate symbol record (see “BLSRESSY” on page 373) in which fields ESSYAS, ESSYLAD, and ESSYDT have been initialized to locate the block.
 - Two or more bytes of binary zeros, indicating that neither of the preceding has been supplied.

Output

1. The requested record is returned.
2. If an equate symbol record was passed, field ESSYD is returned.

BLSRSAPC: Identify Damaged Storage Area

Function: The pointer identifying the damaged area of storage is converted and then inserted in the text portion of the output message. Next the area by which the damaged area was located is likewise converted and inserted in the text portion of another output message.

Modules Called: BLSRMSGGA, BLSRMSGD, BLSUALLO, BLSUFRE1, BLSUPUTO

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Storage address record for the damaged area (see “BLSRSASY” on page 391)
3. Storage address record for the locating area (see “BLSRSASY” on page 391)

Output: None

BLSRSAPU: Add a Storage Address Record

Function: A request parameter list is created to add the EQUATE symbol record. If the RPL is built without any problem, BLSUVSPU is called to actually add the record. If there is any difficulty building the RPL, then a message is issued.

Modules Called: BLSUALLO, BLSUFRE1, BLSUVSCR, BLSUVSPU

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Storage address record (see “BLSRSASY” on page 391)

Output: None

Scan Routines

The program control flow chart showing the modules used for scanning control blocks is presented in Figure 111.

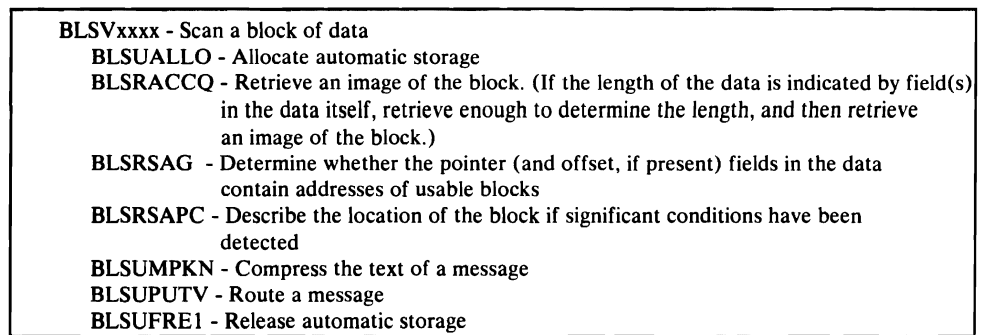


Figure 111. Scan Routines Control Flow

Each scan routine performs a data validation function for one type of data (or group of related types of data). Each receives a common parameter list and is expected to produce comparable output for the data examined:

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Storage address record (see “BLSRSASY” on page 391) describing the block to be scanned.
3. One of three blocks may be supplied:
 - Storage address record (see “BLSRSASY” on page 391) describing the locating block
 - Equate symbol record (see “BLSRESSY” on page 373) describing the block to be scanned
 - A halfword of binary zeroes

Output: Additional map entries may be created in the dump directory data set. Messages are issued for all errors diagnosed.

Modules Called: BLSRACCQ, BLSRSAG, BLSRSAPC, BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUPUTV

Figure 112 provides more information regarding scan routines.

Module	Data Types Scanned
BLSVASCB	STRUCTURE(ASCB)
BLSVASSB	STRUCTURE(ASSB)
BLSVASTE	STRUCTURE(ASTE), STRUCTURE(AST)
BLSVASVT	STRUCTURE(ASVT)
BLSVASXB	STRUCTURE(ASXB)
BLSVCDE	STRUCTURE(CDE), STRUCTURE(CDEMAJOR), STRUCTURE(CDEMINOR), STRUCTURE(LPDE), STRUCTURE(LPDEMAJOR), STRUCTURE(LPDEMINOR)
BLSVCQE	STRUCTURE(CQE)
BLSVCSD	STRUCTURE(CSD)
BLSVCVT	STRUCTURE(CVT)
BLSVCVT2	STRUCTURE(CVTXTNT2)
BLSVGDA	STRUCTURE(GDA)
BLSVGV	STRUCTURE(ISGGVT)
BLSVGVTX	STRUCTURE(ISGGVTX)
BLSVLCCA	STRUCTURE(LCCA)
BLSVLCCV	STRUCTURE(LCCAVT)
BLSVLDA	STRUCTURE(LDA)
BLSVMOD	MODULE(pgmname)
BLSVORE	STRUCTURE(ORE)
BLSVPCCA	STRUCTURE(PCCA)
BLSVPCCV	STRUCTURE(PCCAVT)
BLSVPGTE	STRUCTURE(PGTE), STRUCTURE(PGT)
BLSVPSA	STRUCTURE(PSA)
BLSVPVT	STRUCTURE(PVT)
BLSVQCB	STRUCTURE(ISGQCB)
BLSVQEL	STRUCTURE(ISGQEL)
BLSVQHT	STRUCTURE(ISGQHT)
BLSVRB	STRUCTURE(RB), STRUCTURE(IRB), STRUCTURE(PRB), STRUCTURE(SIRB), STRUCTURE(SVRB), STRUCTURE(TIRB)
BLSVRSV	STRUCTURE(ISGRSV)
BLSVRTCT	STRUCTURE(RTCT)
BLSVSCVT	STRUCTURE(SCVT)
BLSVSGTE	STRUCTURE(SGTE), STRUCTURE(SGT)
BLSVTCB	STRUCTURE(TCB)
BLSVUCB	STRUCTURE(UCB), STRUCTURE(UCBCTC), STRUCTURE(UCBDA), STRUCTURE(UCBGFX), STRUCTURE(UCBTAP), STRUCTURE(UCBTP), STRUCTURE(UCBUR), STRUCTURE(UCB3270)
BLSVUCM	STRUCTURE(UCM)
BLSVWQE	STRUCTURE(WQE)
BLSVXTLS	STRUCTURE(XTLST)

Figure 112. Summary of Scan Routines

Dump Access

There are four classes of storage access service routines:

- The Buffer manager
- Block read routines
- Storage access routines
- PRDMP exit services

The Buffer Manager

BLSRACCB: Assign Dump Buffer

Function: Accept a block address as an argument. Return the index of a buffer for the designated block and a return code which indicates whether the buffer already contains the block.

Input:

1. Active task variable (see "BLSUZZ2" on page 406)
2. A block address
3. A fullword index

Output: The fullword index is set to reference the buffer assigned.

Block Read Routines

There are several block read routines. All receive comparable parameter lists and all retrieve a dump block into a buffer, the address of which is returned to the caller.

Module	Description (Modules Called)
BLSLFINB	Treats the address of the block as the virtual storage address of a buffer maintained by the dump display reporter. ³ (BLSUALLO, BLSUFREI)
BLSRACCA	Treats the address of the block as a BDAM relative record number. (BLSRACCB, BLSUALLO, BLSUFREI)
BLSRACCC	Treats the address of the block as a BSAM relative record number for a data set on tape. (BLSRACCB, BLSRDUVS, BLSUALLO, BLSUFREI, BLSUPGME)
BLSRACCP	Treats the address of the block as a buffer number (0 or 1) of a buffer for active main storage data. (No modules are called.)
BLSRACCT	Treats the address of the block as a relative track address (TTR). (BLSRACCB, BLSUALLO, BLSUFREI)

Input:

1. Active task variable (see "BLSUZZ2" on page 406)
2. Address of the block
3. Fullword pointer

Output: The fullword pointer is set to the address of the buffer containing the block.

³ BLSLFINB is a secondary entry point in module BLSLFIND. It is not a separate IPCS module.

Storage Access Routines

These are storage access routines: BLSRACC, BLSRACCN, and BLSRACCQ. Each storage access routine receives a similar parameter list, and each moves dump data into a buffer supplied by the caller. The storage access routines are summarized in Figure 113 .

Module	Function Performed	Modules Called
BLSRACC	Accesses storage described by an equate symbol record. If the requested storage is not all retrieved, message BLS18100I is generated.	BLSRACCQ, BLSRMSG, BLSRSSC, BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUPUTV
BLSRACCN	Accesses the definition of a symbol and retrieves that portion of the described storage of interest to the caller. If the request cannot be satisfied, a message is generated.	BLSRACC, BLSRESGU, BLSUALLO, BLSUFRE1
BLSRACCQ	Accesses storage described by an equate symbol record.	BLSLFINB, BLSRACCA, BLSRACCC, BLSRACCP, BLSRACCT, BLSRESCK, BLSRRAGE, BLSUALLO, BLSUFRE1

Figure 113. Summary of Storage Access Routines

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A symbol table record describing the storage to be retrieved. For BLSRACCN only fields ESSYSYM and ESSYD need to be filled in.
3. A buffer large enough to accommodate the requested storage.

Output

1. Symbol table record fields ESSYMAD, ESSYKEY, ESSYFS, and ESSYABS are completed.
2. The buffer is filled. The initial physical byte of the buffer receives an image of the initial physical byte of the requested storage. If, for example, an image of an MVS/XA task control block (TCB) is requested including the 32-byte floating-point save area prefix, the initial byte of the buffer receives the first byte of the floating point save area prefix, not the byte at the logical origin of the TCB.

PRDMP Exit Services

There are two PRDMP exit service routines, BLSRACCI and BLSRACCL. Both exit service routines receive the same parameters and both retrieve data into a buffer whose address is returned to the caller:

Input

Reg	Contents
0	As required for the PRDMP storage access service requested.
1	The active task variable (see “BLSUZZ2” on page 406)
13	The address of a 72-byte register save area.
14	The return address to the exit.
15	The address of the entry point being called.

Output

Reg	Contents
0	As required for the PRDMP storage access service requested.
15	A return code:
	Code Meaning
0	Request satisfied
4	Request not satisfied

Figure 114 summarizes the processing of these routines.

Module	Function Performed	Modules Called
BLSRACCI	Permits MVS/XA PRDMP exits to request storage during either AMODE(24) or AMODE(31) execution. BLSRACCI: 1. Uses the register save area passed by the caller. 2. Acquires storage for a save area below 2²⁴ and chains the caller's save area to it, 3. Passes the request to BLSRACCL, invoking BLSRACCL in AMODE(24), and 4. Returns the results produced by BLSRACCL to its caller.	BLSRACCL, BLSUALLO, BLSUFRE1
BLSRACCL	Processes the request for acces to dump data.	BLSRACCQ, BLSUALLO, BLSUFRE1

Figure 114. Summary of PRDMP Exit Storage Access Routines

Dump Initialization

The program control flow chart listing the modules used by the dump initialization function is presented in Figure 115 .

```

BLSRDUAL - Allocate dump
  BLSUALLO - Allocate automatic storage
  BLSRMDSN - Format default dump name
  BLSRZZ6C - Create ZZ6
  BLSUDYNA - Allocate dataset
  BLSUPGME - Call non-resident IPCS program
    BLSRRDGE - Get existing dump record
  BLSUPGME - Call non-resident IPCS program
    BLSRPHGE - Get dump volume data
  BLSUPGME - Call non-resident IPCS program
    BLSRM154 - Transmit message BLS18154I
  BLSUPGME - Call non-resident IPCS program
    BLSRM185 - Transmit message BLS18185I
  BLSUPGMU - Load dump table
    BLSRDUT1 - MVS dump
    BLSRDUT2 - RECFM=F
    BLSRDUT3 - RECFM=U
    BLSRDUT4 - PDS directory
    BLSRDUT5 - PDS member
  BLSUPGME - Call non-resident IPCS program
    BLSRRDPU - Add dump record
  IKJEFT25 - Display time stamp
  BLSUPGMS - Call sequential setup routines
    BLSRDUIN - MVS dump
    BLSRDU12 - RECFM=F
    BLSRDU13 - RECFM=U, PDS directory, or PDS member
  BLSUPGME - Call non-resident IPCS program
    BLSRPHAR - Store dump volume data
  BLSUPGMS - Call direct open routines
    BLSRDUOP - MVS dump
    BLSRDUO2 - RECFM=F
    BLSRDUO3 - RECFM=U
    BLSRDUO4 - PDS directory
    BLSRDUO5 - PDS member
  BLSUPGMS - Call direct setup routine
    BLSRDUS1 - MVS dump
  BLSUPGME - Call non-resident IPCS program
    BLSRRDAR - Add/replace RD record
  BLSUMPK1 - Compress message text
  BLSUTRMV - Transmit message on error
  BLSUPGMS - Call non-resident IPCS routine
    BLSRZZ3U - Update address space
  BLSUPGMS - Call non-resident IPCS routine
    BLSRDUDR - Dropdump service routine
  BLSRDUCC - CLOSE/FREE dump DSN
  BLSUFRE1 - Free automatic storage

BLSRDUCC - Close named dump
  BLSUALLO - Allocate automatic storage
  BLSRDUCD - Close dump identified by BLSRZZ6
  BLSUFRE1 - Free automatic storage

BLSRDUCD - Close dump identified by BLSRZZ6
  BLSUDYNA - Free dump data set
  BLSUPGMD - Delete dump-related entry points
  
```

Figure 115 (Part 1 of 4). Dump Initialization Control Flow

BLSRDUCK - Dump initialization
 BLSUALLO - Allocate automatic storage
 BLSUTRMV - Transmit environmental error message
 BLSUPGMS - Call non-resident program
 BLSRDUAL - Allocate dump
 BLSRDUCD - Close dump
 BLSUFRE1 - Free automatic storage

 BLSRDUCL - Close all dumps
 BLSRDUCD - Close dump identified by BLSRZZ6

 BLSRDUIC - Process CPU status record
 BLSUALLO - Allocate automatic storage
 BLSUMPK1 - Compress message text
 BLSRTRMV - Transmit a message
 BLSUFRE1 - Free automatic storage

 BLSRDUIH - Process header record
 BLSUALLO - Allocate automatic storage
 BLSRESAR - Store symbol TITLE for the dump
 BLSUPGME - Call non-resident IPCS program
 BLSRPHAR - Record dump reuse data
 BLSRTRMV - Transmit a message
 BLSUFRE1 - Free automatic storage

 BLSRDUIN - MVS dump sequential setup
 BLSUALLO - Allocate automatic storage
 BLSUPGMU - Load EOVS exit—BLSRDUVE
 BLSUPGME - Call non-resident IPCS program
 BLSRM154 - Transmit message BLS18154I
 BLSRDUVE - EOVS exit
 BLSRDUIH - Process header record
 BLSRDUIC - Process CPU status record
 BLSRDUIS - Process storage record
 BLSRRAAR - Record location of storage
 BLSUPGMD - Delete EOVS exit—BLSRDUVE
 BLSUFRE1 - Free automatic storage

 BLSRDUIS - Process storage record
 BLSUALLO - Allocate automatic storage
 BLSUPGMU - Load BLS DVT
 BLSFCN00 - Confirm use of summary dump storage
 BLSRRAAR - Describe summary dump storage
 BLSRSAGS - Map data areas
 BLSRTRMV - Transmit a message
 BLSUFRE1 - Free automatic storage

 BLSRDUI2 - RECFM=F sequential setup
 BLSUALLO - Allocate automatic storage
 BLSUPGMU - Load EOVS exit—BLSRDUVE
 BLSUPGME - Call non-resident IPCS program
 BLSRM154 - Transmit message BLS18154I
 BLSRDUVE - EOVS exit
 BLSUPGMD - Delete EOVS exit—BLSRDUVE
 BLSUFRE1 - Free automatic storage

Figure 115 (Part 2 of 4). Dump Initialization Control Flow

BLSRDUI3 - RECFM=U sequential setup
 BLSUALLO - Allocate automatic storage
 BLSUVSCR - Create a VSAM RPL
 BLSUPGMS - Call non-resident IPCS program
 BLSRDUO3 - Open RECFM=U dump
 BLSUPGMU - Load EOVS exit—BLSRDUVE
 BLSRACCB - Assign dump buffer
 BLSRDUVE - EOVS exit
 BLSRRAAR - Update RA records
 BLSUVSMR - Modify a VSAM RPL
 BLSUVSPU - Write RB and RC records
 BLSUPGMD - Delete EOVS exit—BLSRDUVE
 BLSUFRE1 - Free automatic storage

BLSRDUOP - Open MVS/XA Dump Data Set
 BLSUALLO - Allocate automatic storage
 BLSUVSCR - Create a VSAM RPL
 BLSUPGMU - Load record access routine—BLSRACCX
 BLSRACCA - Read record from DASD
 BLSRACCC - Read record from tape
 BLSRACCP - Access active main storage
 BLSRDUBF - Build buffer pool
 BLSUPGME - Call non-resident IPCS program
 BLSRPHGE - Get dump reuse data
 BLSRRAGE - Locate header record
 BLSRACCX - Read header record
 BLSUPGME - Call non-resident IPCS program
 BLSRM185 - Transmit BLS18185I
 BLSUPGME - Call non-resident IPCS program
 BLSRM154 - Transmit BLS18154I
 BLSUFRE1 - Free automatic storage

BLSRDUO2 - Open RECFM=F Data Set
 BLSUALLO - Allocate automatic storage
 BLSUPGMU - Load record access routine—BLSRACCX
 BLSRACCA - Read record from DASD
 BLSRACCC - Read record from tape
 BLSUPGME - Call non-resident IPCS program
 BLSRM154 - Transmit BLS18154I
 BLSRDUBF - Build buffer pool
 BLSUFRE1 - Free automatic storage

BLSRDUO3 - Open RECFM=U Data Set
 BLSUALLO - Allocate automatic storage
 BLSUPGMU - Load record access routine—BLSRACCX
 BLSRACCC - Read record from tape
 BLSRACCT - Read record from DASD
 BLSUPGME - Call non-resident IPCS program
 BLSRM154 - Transmit BLS18154I
 BLSRDUBF - Build buffer pool
 BLSUFRE1 - Free automatic storage

BLSRDUO4 - Open PDS Directory
 BLSUALLO - Allocate automatic storage
 BLSUPGMU - Load record access routine—BLSRACCX
 BLSRACCT - Read record from DASD
 BLSUPGME - Call non-resident IPCS program
 BLSRM154 - Transmit BLS18154I
 BLSRDUBF - Build buffer pool
 BLSUFRE1 - Free automatic storage

Figure 115 (Part 3 of 4). Dump Initialization Control Flow

BLSRDUO5 - Open PDS Member
BLSUALLO - Allocate automatic storage
BLSUPGMU - Load record access routine—BLSRACCX
BLSRACCT - Read record from DASD
BLSUPGME - Call non-resident IPCS program
BLSRM154 - Transmit BLS18154I
BLSRRBGE - Verify TTR of member
BLSUPGME - Call non-resident IPCS program
BLSRM185 - Transmit BLS18185I
BLSUTRMV - Transmit BLS18190I
BLSRDUBF - Build buffer pool
BLSUFRE1 - Free automatic storage
BLSRDUSO - Open PDS Member
BLSUALLO - Allocate automatic storage
BLSRDUSE - Resolve dump source
BLSRDUCK - Prepare dump for use
BLSUFRE1 - Free automatic storage
BLSRDUS1 - MVS/XA Dump Direct Setup
BLSUALLO - Allocate automatic storage
BLSRESGU - Locate key MVS/XA data areas
BLSRRAGE - Locate storage in dump
BLSUFRE1 - Free automatic storage
BLSRDUVE - EOVS Exit for dumps
BLSUPGME - Call non-resident IPCS program
BLSRPHGE - Get dump volume data
BLSUPGME - Call non-resident IPCS program
BLSRPHAR - Store dump volume data
BLSRDUVS - Switch Dump Volumes
BLSUPGME - Call non-resident IPCS program
BLSRPHGE - Get dump volume data

Figure 115 (Part 4 of 4). Dump Initialization Control Flow

Unless otherwise stated subroutines in this group receive control with the address of the active task variable (see “BLSUZZ2” on page 406) in register 1 and return no output directly to the caller.

BLSRDUAL: Allocate Dump Data Set

Function: BLSRDUAL receives control from BLSRDUCK prior to the execution of the first IPCS subcommand that requires a dump data set during its processing. It allocates, opens, and, if necessary, performs dump initialization relative to the dump.

Modules Called: BLSRDUCC, BLSRDUDR, BLSRDUIN, BLSRDUI2, BLSRDUI3, BLSRDUOP, BLSRDUO2, BLSRDUO3, BLSRDUO4, BLSRDUO5, BLSRDUS1, BLSRMDSN, BLSRM154, BLSRM185, BLSRPHAR, BLSRPHGE, BLSRRDAR, BLSRRDGE, BLSRRDPU, BLSRZZ3U, BLSRZZ6C, BLSUALLO, BLSUDYNA, BLSUFRE1, BLSUMDSN, BLSUMPK1, BLSUPGME, BLSUPGMS, BLSUPGMU, BLSUTRMV

BLSRDUBF: Build Dump Buffer Pool

Function: Build a pool of dump buffers.

Modules Called: BLSUALLO, BLSUFRE1, BLSUTRMV

BLSRDUCC: Close/Free a Dump Data Set

Function: This module determines whether a dump data set is currently open. If it is, it closes the data set, frees the file with which it is associated, and releases the dump data set control block (see “BLSRZZ6” on page 400) and buffer pool associated with it.

Modules Called: BLSRDUCD, BLSUALLO, BLSUFRE1

BLSRDUCD: Close/Free Dump DSN

Function: Determine whether a dump data set is currently open. If it is, close the data set, free the file with which it is associated, and release the dump data set control block (see “BLSRZZ6” on page 400) and buffer pool associated with it.

Modules Called: BLSUDYNA, BLSUPGMD

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Dump control block (see “BLSRZZ6” on page 400)

Output: None

BLSRDUCK: Check Dump Data Set

Function: This module ensures that a dump data set has been opened and made available prior to the execution of IPCS code which requires access to a dump.

Modules Called: BLSRDUAL, BLSRDUCD, BLSUALLO, BLSUFRE1, BLSUPGMS, BLSUTRMV

BLSRDUCL: Close/Free all Dump Data Sets

Function: It determines whether dump data sets are currently open. If they are, BLSRDUCL closes the data sets, frees the files with which they are associated, and releases the dump data set control blocks (see “BLSRZZ6” on page 400) and buffer pools associated with them.

Modules Called: BLSRDUCD

BLSRDUIC: MVS/XA Dump Initialization - CPU Status Record

Function: BLSRDUIC receives control each time a CPU status record is encountered during dump initialization. BLSRDUIC establishes the preferred CPU address for the dump and informs the debugger that the status information is available. Warning flags placed in the dump by AMDSADMP are examined, and the warning(s) are forwarded to the debugger.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. MVS/XA dump CPU status record

Output: None.

BLSRDUIH: MVS/XA Dump Initialization - Header Record

Function: This module receives control when a dump header record has been read during dump initialization. It determines whether a non-blank title was supplied. If a title was supplied, it transmits message BLS18124I and establishes a ‘title’ entry in the symbol table. Otherwise, it transmits message BLS18127I.

Modules Called: BLSRESAR, BLSRPHAR, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. MVS/XA dump header record

Output: None.

BLSRDUIH: MVS/XA Dump Sequential Setup

Function: BLSRDUIH reads all records in a dump data set. It writes dump directory address space records, which captures the same information recorded in the initial eight bytes of the respective dump data set records. It establishes IPCS defaults that pertain to the dump data set.

Modules Called: BLSRDUVE, BLSRDUIC, BLSRDUIH, BLSRDUIS, BLSRM154, BLSRRAAR, BLSUALLO, BLSUFRE1, BLSUPGMD, BLSUPGME

BLSRDUIS: MVS/XA Dump Initialization - Storage Record

Function: Each dump storage record read during dump initialization is passed to BLSRDUIS for inspection and appropriate processing. BLSRDUIS processes summary dump logical records, and records the storage they contain for preferred use by IPCS storage access during dump processing. BLSRDUIS is also concerned with detecting and flagging the presence of absolute storage records in a dump.

Modules Called: BLSFCN00, BLSRRAAR, BLSRSAGS, BLSUALLO, BLSUFRE1, BLSUPGMU, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. MVS/XA dump storage record
3. A 128-byte work area set to zeroes prior to the first call to BLSRDUIS and made available each time BLSRDUIS is called. BLSRDUIS records interrecord relationships between summary dump records in this structure.

Output: None.

BLSRDUI2: RECFM=F Sequential Setup

Function: After an uninitialized RECFM=F data set has been successfully allocated, open it for QSAM locate mode input, read to determine the number of records, and record that number. Close the QSAM DCB.

Modules Called: BLSRDUVE, BLSRM154, BLSUALLO, BLSUFRE1, BLSUPGMD, BLSUPGME, BLSUPGMU

BLSRDUI3: RECFM=U Sequential Setup

Function: After a RECFM=U data set has been successfully allocated, open it for BSAM locate mode input. Scan it for initialization, and close it.

Modules Called: BLSRACCB, BLSRDUVE, BLSRDUO3, BLSRRAAR, BLSRVSCR, BLSUALLO, BLSUFRE1, BLSUPGMD, BLSUPGMS, BLSUPGMU, BLSUVSMR, BLSUVSPU

BLSRDUOP: Open MVS/XA Dump Data Set

Function: BLSRDUOP receives control after a dump data set has been successfully allocated. BLSRDUOP determines whether initialization has been performed for the dump. If it has not been performed, the dump data set is opened for QSAM. QSAM LOCATE mode input and BLSRDUIN are called to perform initialization. If initialization has been (or can be) successfully completed, the dump data set is opened for BDAM relative record input, and a buffer pool is acquired and initialized. The buffer pool is managed throughout processing by IPCS dump management code. MVS/XA data management plays no role.

Modules Called: BLSRACCA, BLSRACCC, BLSRDUBF, BLSRM154, BLSRM185, BLSRPHGE, BLSRRAAR, BLSRRAGE, BLSRSPIE, BLSUALLO, BLSUFRE1, BLSUPGMC, BLSUPGME, BLSUPGMU, BLSUTRMV

BLSRDUO2: Open RECFM=F Data Set

Function: Open a RECFM=F data set for BDAM relative record input, acquire and initialize a buffer pool (managed by IPCS dump management).

Modules Called: BLSRACCA, BLSRACC, BLSRDUBF, BLSRM154, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUPGMU

BLSRDUO3: Open RECFM=U Data Set

Function: Open a RECFM=U data set for BSAM input, and acquire a buffer pool (managed by IPCS dump management).

Modules Called: BLSRACCC, BLSRACCT, BLSRDUBF, BLSRM154, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUPGMU

BLSRDU04: Open PDS Directory

Function: Open a PDS directory for BSAM input, acquire and initialize a buffer pool (managed by IPCS dump management).

Modules Called: BLSRACCT, BLSRDUBF, BLSRM154, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUPGMU

BLSRDU05: Open PDS Member

Function: Open a PDS member for BPAM input, acquire and initialize a buffer pool (managed by IPCS dump management).

Modules Called: BLSRACCT, BLSRDUBF, BLSRM154, BLSRM185, BLSRRBGE, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUPGMU, BLSUTRMV

BLSRDUSE: Record Dump Source

Function: BLSRDUSE processes PDE information supplied by IKJPARS to identify the dump source. BLSRDUSE then records the identity of the designated dump in the active task variable.

Modules Called: None

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. PDL containing dump source PDEs.
3. Offset table indicating the offsets of each dump source PDE within the PDL.

Output: An updated active task variable (see “BLSUZZ2” on page 406)

BLSRDUSO: Access Dump Source

Function: BLSRDUSO processes PDE information supplied by IKJPARS to identify the dump source. BLSRDUSO then prepares for the use of the designated dump data set.

Modules Called: BLSRDUCK, BLSRDUSE, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. PDL containing dump source PDEs.
3. Offset table indicating the offsets of each dump source PDE within the PDL.

Output: An updated active task variable (see “BLSUZZ2” on page 406)

BLSRDUS1: Direct Setup for an MVS/XA Dump

Function: BLSRDUS1 initializes the dump directory for an MVS dump.

Modules Called: BLSRESGU, BLSRRAGE, BLSUALLO, BLSUFRE1

Input: Register 1 - addresses the active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSRDUVE: End-of-Volume Exit

Function: BLSRDUVE receives control from data management as an end-of-volume exit during dump initialization. BLSRDUVE establishes or updates a program history record associated with the dump to indicate the range of records stored on the previous volume.

Modules Called: BLSRPHAR, BLSRPHGE, BLSUPGME

Input

Register	Contents
0	The address of a dump control block (see “BLSRZZ6” on page 400)
9	The address of the active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSRDUVS: Volume Switch Service

Function: BLSRDUVS determines which dump volume contains a block. BLSRDUVS uses CLOSE and OPEN subcommand requests to reposition the dump to that volume.

Modules Called: BLSRPHGE, BLSUPGME

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Block number

Output: None

BLSRZZ6C: Create ZZ6

Function: BLSRZZ6C obtains storage for a dump control block and initializes that storage.

Input: Register 1 - addresses the active task variable (see “BLSUZZ2” on page 406)

Output: The dump control block is created and accessible via the ZZ2AZZ6P field.

BLSUDDSN: Set ZZ2AD Using a PDE

Function: BLSUDDSN sets field ZZ2AD using a DSNNAME PDE.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. DSNNAME PDE

Output: Field ZZ2AD has been updated.

Symbol Management

All symbol management routines share a common parameter list:

1. Active task variable (see “BLSUZZ2” on page 406).
2. Equate symbol record (see “BLSRESSY” on page 373)

The routines differ based on:

- The degree to which the equate symbol record must be initialized upon entry.
- The processing performed using the equate symbol record.

There are two groups of symbol management routines:

- Symbol retrieval routines are summarized in Figure 116 on page 295 .
- Symbol table update routines are summarized in Figure 117 on page 295 .

Module	Function Performed	Modules Called
BLSRESCK	Determines whether an equate symbol record (see “BLSRESSY” on page 373) provides a reasonable description of debugging storage. If not, and the TEST option has been requested to activate IPCS internal diagnostic facilities, it abends to permit the source of the improper data to be isolated and the IPCS problem rectified. Otherwise, a modified, proper equate symbol record is substituted, an error condition is signalled to the caller, and processing is continued.	BLSRDATA, BLST02, BLSUALLO, BLSU, FRE1, BLSUTRMV
BLSRESGA	Obtains the definition of a standard symbol. Recognize the use of an alias prior to accessing the symbol table. If no definition can be obtained, transmit a message to the terminal.	BLSRESGU, BLSRSSA, BLSUALLO, BLSUFRE1
BLSRESGC	Retrieves a specified equate symbol record from the dump directory.	BLSUALLO, BLSUFRE1, BLSUVSCR, BLSUVSGE
BLSRESGE	Obtains the definition of a symbol in one of the following ways: - From an active record queue in virtual storage. - From the dump directory. - By dynamic resolution, passing control to a find routine.	BLSRESGC, BLSRESGQ, BLSRESPU, BLSSaaaa, BLSUALLO, BLSUFRE1, BLSUPGMC
BLSRESGQ	Obtains the definition of a symbol from an active record queue in virtual storage. The queue is maintained to avoid recursion when for example, BLSSCVT attempts to define the symbol CVT in virtual storage, preferring that definition to the one in absolute storage. During this resolution process the usable (but less desirable) definition in absolute storage is made available in the active record queue.	None
BLSRESGU	Obtains the definition of a symbol via BLSRESGE. If no definition can be obtained, message BLS18104I is transmitted to the terminal.	BLSRESGE, BLSRSSC, BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUPUTV
BLSRESSA	Checks the validity of a block of storage.	BLSRSAGU, BLSUALLO, BLSUFRE2
BLSRESSR	Checks the validity of a block of storage. Passes the description of the block to the scan process.	BLSRSAGR, BLSUALLO, BLSUFRE1

Figure 116. Symbol Retrieval Routine Summary

Module	Function Performed	Modules Called
BLSRESAR	Stores a record in the symbol table. If necessary, it replaces an existing record.	BLSRESCK, BLSRRDAR, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUVSAR, BLSUVSCR
BLSRESPU	Adds an equate symbol record to the dump directory. The caller ensures that the symbol is not defined prior to calling BLSRESPU.	BLSRESCK, BLSRRDAR, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUVSCR, BLSUVSPU
BLSRESSS	Determines the highest-numbered stack symbol in the symbol table for a dump and adds an equate symbol record to the dump directory for the successor to that symbol.	BLSRESPU, BLSUALLO, BLSUFRE1, BLSUVSCR, BLSUVSEN, BLSUVSGU, BLSUVSMR, BLSUVSPO

Figure 117. Symbol Table Update Routine Summary

Special Symbol Processing

Special symbol processing subroutines are named BLSRSSaa, where “aa” is a two-character suffix.

Primary Subroutines

Primary special symbol processing subroutines are named BLSRSSSa, where “a” is a one-character suffix. There are three primary special symbol processing subroutines. Each receives the same parameter list and produces similar output:

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A 31-character buffer containing a symbol to be edited

Output: The same 31-character buffer containing the edited symbol.

Modules Called:: BLSUALL0, BLSUFRE1, and the routines mentioned in the individual module descriptions.

BLSRSSSA: Converts symbols entered as part of IPCS subcommands to the standard internal format used by IPCS. It uses BLSRSSSP to identify special symbol prefixes and to route control to BLSRSS01, BLSRSS02, BLSRSS03, BLSRSS04, or BLSRSS05.

BLSRSSSC: Converts symbols from the standard internal format to a compact format for use in IPCS displays. It uses BLSRSSSP to identify special symbol prefixes and to route control to BLSRSSC1, BLSRSSC2, BLSRSSC3, BLSRSSC4, or BLSRSSC5.

BLSRSSSP: Identifies special symbol prefixes and routes control to suffix editing subroutines using BLSUPGME. In this process it uses data area “BLSRSST” on page 395 which lists the special symbol prefixes associated with the dump being processed and associates those prefixes with the suffixes they require. Module BLSRSST1 is the only special symbol table at this time. It describes the special symbols for MVS/XA dumps.

Suffix Editing Subroutines

Each suffix editing subroutine receives the same parameter list and produces similar output:

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A 31-character buffer containing a suffix to be edited

Output: The same 31-character buffer containing the edited suffix.

The following suffix editing subroutines are supplied:

Input Editor	Output Editor	Type of Suffix Processed
BLSRSS01	BLSRSC1	ASID suffix
BLSRSS02	BLSRSC2	ASID/COUNT
BLSRSS03	BLSRSC3	CPU address
BLSRSS04	BLSRSC4	Unit address
BLSRSS05	BLSRSC5	AST index

Finding Data Areas

The program control flow chart listing the modules used by the finding data areas function is presented in Figure 118. The 'xxxx' in Figure 118 indicates that all find routines share essentially the same control flow. They also share the same parameter list, generate equivalent output, and call similar modules to accomplish their processing:

BLSxxxx - Find a block of significant data BLSUALLO - Allocate automatic storage BLSRESGU - Locate another block of data which is needed to locate this data BLSRACC - Retrieve the fields which simply point to this data or, in complex cases, are required to compute the address of this data BLSRSAGU - Determine whether the storage located appears to contain usable data (in some circumstances this step is not performed.) BLSUFRE1 - Release automatic storage

Figure 118. Finding Data Areas Control Flow

Modules Called: BLSRACC, BLSRESGU, BLSRSAGU, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see "BLSUZZ2" on page 406).
2. A buffer for an equate symbol record (see "BLSRESSY" on page 373).

All fields but ESSYAS, ESSYLAD, ESSYD, and ESSYR must be initialized. Field ESSYSYM must contain a symbol, and that symbol must be of the form shown for all find routines which can locate more than one block. The symbol is ignored by those find routines which locate only one block, but all IPCS internal callers use the symbol shown.

Output: Fields ESSYAS, ESSYLAD, ESSYD, and ESSYR have been supplied.

Figure 119 on page 298 provides more information regarding these routines.

The find routines shown in Figure 119 on page 298 have been implemented.

The lowercase terms in Figure 119 are defined as follows:

aaaaa	A 5-digit decimal MVS/XA address space identifier (ASID) is used in symbols ASCBaaaaa, ASXBaaaaa, PGTaaaaaqqqqq, SGTaaaaa, and TCBaaaaaqqqqq.
cc	A 2-digit decimal CPU address is used in symbols LCCAcc and PCCAcc.
ddd	A 3-digit hexadecimal device address is used in UCBddd.
ffff	A 4-digit decimal ASN-first-table entry number is used in ASTffff.
pgmname	A program name containing between one and eight characters is used in CDEpgmname, LPDEpgmname, pgmname, and XLpgmname.
qqqqq	A 5-character base-26 number is used in PGTaaaaaqqqqq and TCBaaaaaqqqqq. (The 26 upper case letters, A-Z, are used for digits.)

This naming technique ensures that the symbols for page tables and task control blocks related to a specific address space collate together and collate in a reasonably *natural* order in the symbol table.

Routine	Data Type	Symbol	Preceding-Block.Field
BLSSAFT	STRUCTURE(AFT)	AFT	(control register 14)
BLSSASCB	STRUCTURE(ASCB)	ASCBaaaaa	ASVT.ASVTENTY
BLSSASMV	STRUCTURE(ASMVT)	ASMVT	CVT.CVTASMVT
BLSSASTE	STRUCTURE(ASTE)	ASTffff	AFT.entry
BLSSASVT	STRUCTURE(ASVT)	ASVT	CVT.CVTASVT
BLSSASXB	STRUCTURE(ASXB)	ASXBaaaaa	ASCBaaaaa.ASCBASXB
BLSSCDE	STRUCTURE(CDE)	CDEpgmname	CVT.CVTQLPAQ (cde).CDCHAIN
BLSSCOMM	AREA(COMMON)	COMMON	CVT.CVTSHRVM
BLSSCSD	STRUCTURE(CSD)	CSD	CVT.CVTCSD
BLSSCVT	STRUCTURE(CVT)	CVT	SDUMP header record (psa).FLCCVT2 (psa).FLCCVT (alternate)
BLSSCVT2	STRUCTURE(CVTXTNT2)	CVTXTNT2	CVT.CVTEXT2
BLSSGDA	STRUCTURE(GDA)	GDA	CVT.CVTGDA
BLSSGVT	STRUCTURE(ISGGVT)	ISGGVT	CVT.CVTGVT
BLSSGVTX	STRUCTURE(ISGGVTX)	ISGGVTX	GVT.GVTGVTX
BLSSLCCA	STRUCTURE(LCCA)	LCCAcc	LCCAVT.LCCATccP
BLSSLCCV	STRUCTURE(LCCAVT)	LCCAVT	CVT.CVTLCCAT
BLSSLPA	MODULE(pgmname)	pgmname	XLpgmname.XTLMSBAA XLpgmname.XTLMSBLN CDEpgmname.CDENTPT LPDEpgmname.LPDENTP LPDEpgmname.LPDEMIN LPDEpgmname.LPDEMJNM LPDEpgmname.LPDEXTLN
BLSSLPDE	STRUCTURE(LPDE)	LPDEpgmname	CVT.CVTVMMDI CVT.CVTLPDIA (lpde).LPDECHN (lpde).LPDENAME
BLSPCCA	STRUCTURE(PCCA)	PCCAcc	PSAcc.PSACPUPA PSAcc.PSAPCCAR PCCAVT.PCCATccP
BLSPCCV	STRUCTURE(PCCAVT)	PCCAVT	CVT.CVTGCCAT
BLSPFT	STRUCTURE(PFTE)	PFT	PVT.PVTPFTP PVT.PVTFPFN PVT.PVTLPFN

Figure 119 (Part 1 of 2). Summary of Find Routines

Routine	Data Type	Symbol	Preceding-Block.Field
BLSSPGTE	STRUCTURE(PGTE)	PGTaaaaaqqqqq	SGTaaaaa.SGTPAM SGTaaaaa.SGTPTL SGTaaaaa.SGTSTE
BLSSPRIV	AREA(PRIVATE)	PRIVATE	CVT.CVTNUCB COMMON (common area address)
BLSSPRIX	AREA(PRIVATEX)	PRIVATEX	GDA.GDAEPVT
BLSSPSA	STRUCTURE(PSA)	PSAcc	PCCAcc.PCCAPSAR
BLSSPVT	STRUCTURE(PVT)	PVT	CVT.CVTPVIP
BLSSQHT	STRUCTURE(ISGQHT)	ISGQHTG ISGQHTL	GVTX.GVTXGQHT GVTX.GVTXLQHT
BLSSRSV	STRUCTURE(ISGRSV)	ISGRSV	GVTX.GVTXBRSV
BLSSRTCT	STRUCTURE(RTCT)	RTCT	CVT.CVTRTMCT
BLSSSCVT	STRUCTURE(SCVT)	SCVT	CVT.CVTABEND
BLSSSGTE	STRUCTURE(SGTE)	SGTaaaaa	PSAcc.PSASTOR ASCBaaaaa.ASCBSTOR ASCBaaaaa.ASCBSEQN
BLSSTCB	STRUCTURE(TCB)	TCBaaaaaqqqqq	ASXBaaaaa.ASXBFTCB ASXBaaaaa.ASXBTCBS TCBaaaaarrrrr.TCBTCBP (rrrrr=qqqqq-1)
BLSSUCB	STRUCTURE(UCB)	UCBddd	CVT.CVTUCBA UCBxxx.UCBNXUCB (xxx¬=ddd)
BLSSUCM	STRUCTURE(UCM)	UCM	CVT.CVTCUCB
BLSSXTLS	STRUCTURE(XTLST)	XLpgmname	CDEpgmname.CDMIN CDEpgmname.CDXLMSP CDEpgmname.CDNIC CDEpgmname.CDXLE

Figure 119 (Part 2 of 2). Summary of Find Routines

Address Space Mapping

The program control flow chart listing the modules used by the address space mapping function is presented in Figure 120 on page 300 .

BLSRRAAR	- Merge information regarding data contained in a dumped address space with any already stored in the dump directory
BLSUALLO	- Allocate automatic storage
BLSUVSCR	- Create two VSAM RPLs
BLSUVSPO	- Identify first (next) record of interest
BLSUVSRE	- Erase an address space record
BLSUVSGU	- Retrieve first (next) record
IKJEFF19	- Diagnose VSAM I/O errors
BLSUVSPU	- Add or modify an address space record
BLSUVSMR	- Modify RPL for final, short record
BLSUVSEN	- Terminate sequential scan
BLSUFRE1	- Release automatic storage
BLSRRACI	- Map active main storage
BLSRSPIE	- Intercept storage access exceptions
BLSRRAGE	- Retrieve an address space record
BLSUALLO	- Allocate automatic storage
BLSUVSCR	- Create a VSAM RPL
BLSUVSGU	- Retrieve a dump directory record
BLSUVSEN	- Terminate VSAM string
BLSRRBGE	- Locate TTR for block
BLSRRCGE	- Verify that TTR is used
BLSRRACI	- Obtain active storage
BLSRRACR	- Extend the definition of a REAL address space
BLSRRACV	- Extend the definition of an MVS/XA virtual address space
BLSUFRE1	- Release automatic storage
BLSRRACR	- Extend the definition of a REAL address space
BLSUALLO	- Allocate automatic storage
BLSRESGU	- Locate the PSA for the CPU
BLSRRAGE	- Retrieve ABSOLUTE storage address space description
BLSRRAAR	- Merge REAL storage address space description with what is already known
BLSUFRE1	- Release automatic storage
BLSRRACV	- Extend the definition of an MVS/XA virtual address space
BLSUALLO	- Allocate automatic storage
BLSRRAGE	- Retrieve common area storage address space description from the preferred address space on behalf of a secondary address space
BLSRESGU	- Locate the segment table for the address space, and the page table for the segment
BLSRZALC	- Compute segment number suffix
BLSRACCN	- Retrieve page table entry for the page
BLSRESGU	- Locate the page frame table
BLSRACCQ	- Retrieve page frame table entry for the page which last contained an image of the page
BLSRRAGE	- Retrieve REAL storage address space description for the page
BLSRRAAR	- Merge MVS/XA virtual storage address space description with what is already known
BLSUFRE1	- Release automatic storage

Figure 120. Address Space Mapping Control Flow

BLSRRAAR: Add or Replace Address Space Record Data

Function: BLSRRAAR accepts information from IPCS analysis routines that describe the presence or absence of storage for part of an address space. If the storage is present, the disk addresses that contain the information are recorded to speed any subsequent request for access to the storage.

Modules Called: BLSUALLO, BLSUFRE1, BLSUVSCR, BLSUVSEN, BLSUVSRE, BLSUVSGU, BLSUVSMR, BLSUVSPO, BLSUVSPU, IKJEFF19

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. RAPA - An address space record (see “BLSRRASY” on page 385). The RAPA structure provided as an input parameter must be merged with any existing information regarding the address space stored in the dump directory.

Output: One or more address space records are written to the dump directory, merging the parameter data with that previously stored.

***BLSRRACI:* Describe MVS/XA Active Main Storage**

Function: When BLSRRAGE gets a request to describe one or more bytes of MVS/XA storage from the address space in which IPCS is executing, BLSRRAGE passes control to BLSRRACI to obtain the requested information. BLSRRACI determines whether the storage is available by copying its contents into a buffer.

Modules Called: BLSRSPIE

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. A dump directory address space record (see “BLSRRASY” on page 385). Field RASYAS must contain the designated address space, and field RASYFAD must contain the address that BLSRRACI uses to copy the information into.

Output: The second parameter is changed to provide the information requested in the first array entry.

***BLSRRACR*: Describe MVS/XA Real Address Space**

Function: When BLSRRAGE is called requesting the description of one or more bytes of dumped MVS/XA real storage, it initially attempts to satisfy the request using an address space record stored in the dump directory. If no information about the storage is available, BLSRRACR is called to provide the requested information and, optionally, to store the information developed into the dump directory to satisfy any subsequent requests regarding the same storage.

Modules Called: BLSRESGU, BLSRRAAR, BLSRRAGE, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. A dump directory address space record (see “BLSRRASY” on page 385). Field RASYAS must be filled in with the description of the designated address space, and field RASYFAD must be filled in with the address regarding which BLSRRACR is to produce information.

Output

1. Information collected that would be costly to recreate is written to the dump directory using BLSRRAAR.
2. The second parameter is changed to provide the information requested in the first array entry.

BLSRRACV: Build MVS/XA Virtual Address Space Description

Function: When BLSRRAGE is called requesting the description of one or more bytes of dumped MVS/XA virtual storage, it initially attempts to satisfy the request using an address space record stored in the dump directory. If no information about the storage is available, BLSRRACV is called to provide the requested information and, optionally, to store the information developed into the dump directory to satisfy any subsequent requests regarding the same storage. Thus, for purposes of this processing, BLSRRAGE can be considered to function very much like an MVS/XA central processor, using the dump directory as the equivalent of a translation lookaside buffer, falling back on BLSRRACV to access the segment table, page table, and real storage management tables required to resolve information regarding the requested storage.

Modules Called: BLSRACCN, BLSRACCQ, BLSRESGU, BLSRRAAR, BLSRRAGE, BLSRZALC, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. A dump directory address space record (see “BLSRRASY” on page 385). Field RASYAS must be filled in with the description of the designated address space, and field RASYFAD must be filled in with the address regarding which BLSRRACV is to produce information.

Output

1. Information collected that would be costly to recreate is written to the dump directory using BLSRRAAR.
2. The second parameter is changed to provide the information requested in the first array entry.

BLSRRAGE: Retrieve Address Space Description

Function: This module determines whether a particular byte of an address space is available.

- If the byte is available, it describes where that byte and any others in the immediate vicinity can be obtained.
- If the byte is not available, it indicates that fact and the fact, if already known, that subsequent bytes are or are not available.

If no information is currently recorded about the byte, and the address space is real storage or MVS/XA virtual storage, it calls BLSRRACR or BLSRRACV to extend the definition of the address space to include the designated byte. Otherwise, it

assumes the convention that only available storage is recorded in the dump directory, and it generates a description of the byte as not available.

Modules Called: BLSRRACI, BLSRRACR, BLSRRACV, BLSRRBGE, BLSRRCGE, BLSUALLO, BLSUFRE1, BLSUVSCR, BLSUVSEN, BLSUVSGU

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. A buffer for a dump directory address space record (see “BLSRRASY” on page 385) large enough to contain a minimum size record. Fields RASYAS and RASYFAD must indicate the address space and the byte address of interest, respectively.

Output: The address space record contains the information requested.

BLSRSPIE: ESPIE exit

Function: BLSRSPIE accepts protection exceptions encountered during execution of an MVCL instruction that accesses active main storage for IPCS problem analysis. If any other interruption occurs, BLSRSPIE terminates abnormally with a system ABEND code of X'0C4'.

Modules Called: None

Input: Register 1 - addresses an extended program interruption element

Output: If the protection exception was expected, the PSW is adjusted to resume after instruction BLSRMVCL in module BLSRRACI.

Address Resolution

The program control flow chart listing the modules used by the address resolution function is presented in Figure 121.

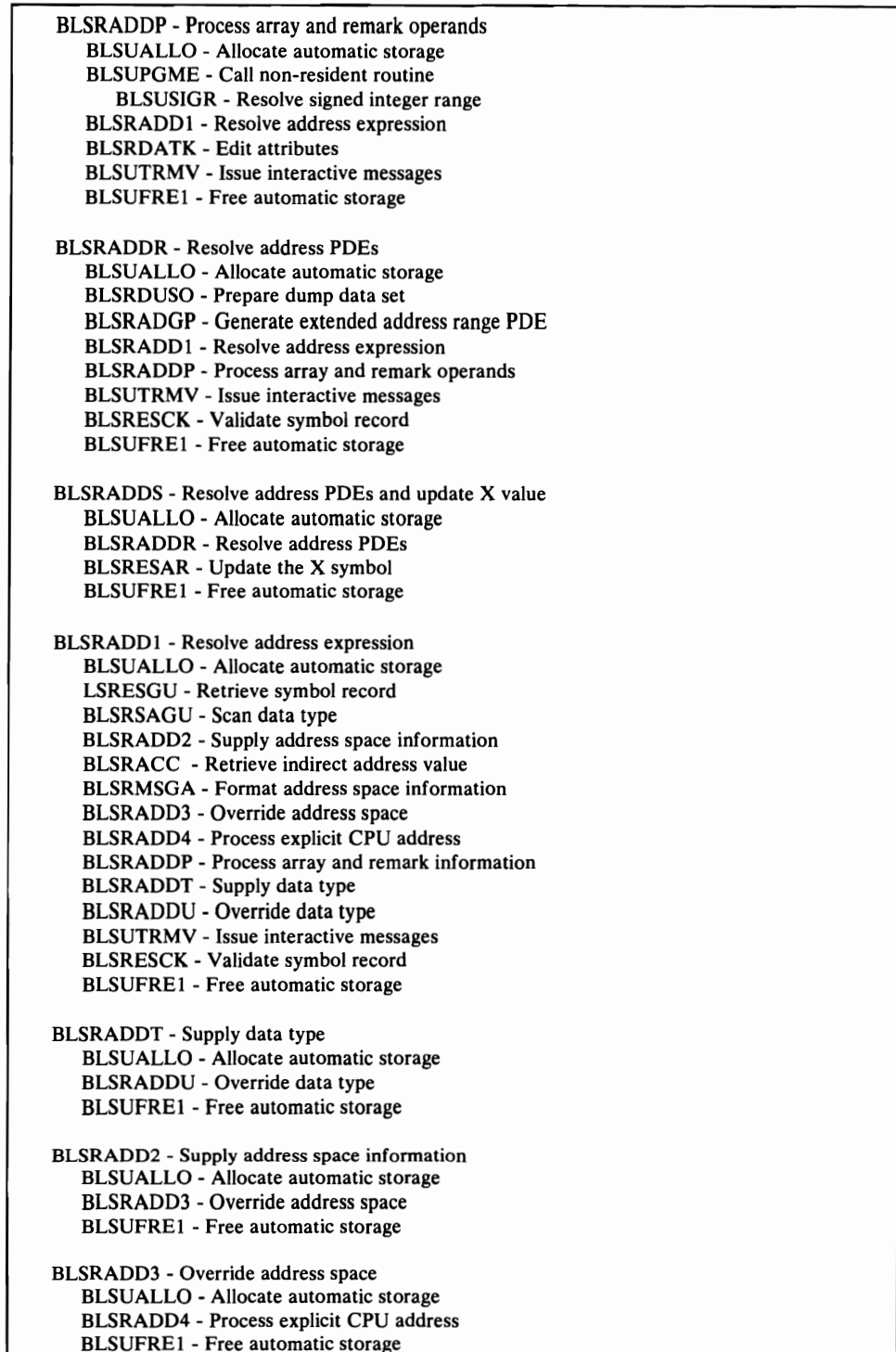


Figure 121. Address Resolution Control Flow

BLSRADDP: Resolve Address PDEs

Function: This module processes array and remark information as supplied in the subcommand or defaulted.

1. It processes the entries, dimension, multiple, and scalar keywords. For array processing, it ensures that the upper bound does not exceed limits. If so, then it issues a message, adjusts the length, and converts to scalar.
2. If a pointer or unsigned data type has a length greater than 4, or a signed data type has a length not a multiple of two, then it changes the data type to AREA.
3. It processes any remark information that is supplied on the subcommand.

Modules Called: BLSRDATK, BLSUALLO, BLSUFRE1, BLSUPGME, BLSUSIGR, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. Equate symbol record (see “BLSRESSY” on page 373)
3. PD, PDL (Parameter Descriptor List).
4. PXAO, PDE (Parameter Descriptor Element Array).

Output: Additional information is completed in the equate symbol record for the functions described above. This includes ‘array’ and ‘remark’ information.

BLSRADDR: Resolve Address PDEs

Function: BLSRADDR processes the address PDEs that are returned from the TSO parse routine.

1. It calls are made to BLSRADD1 to process an address PDE. If a range is specified, then BLSRADD1 is called a second time. Also, if a range is specified, then checking is done to ensure that the array or scalar descriptions are consistent.
2. It calls BLSRADDP to process any array or remark information.
3. Finally the storage description (symbol record) is checked for validity by a call to BLSRESCK.

Modules Called: BLSRADDP, BLSRADD1, BLSRADGP, BLSRDUSO, BLSRESCK, BLSUALLO, BLSUFRE1, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Equate symbol record (see “BLSRESSY” on page 373)
3. PD, PDL
4. PXAO, PDE
5. PDAD, PDE for address range.

Output: Additional information is completed in the equate symbol record for the functions described above.

BLSRADDs: Resolve Address PDEs

Function: This module processes PDE information as supplied by the TSO parse routine. This routine is called by those routines that want to insure that the 'X' symbol value is updated upon successful completion of resolving the address PDE. The address PDE information is processed by calling the BLSADDR routine.

Modules Called: BLSRADDR, BLSRESAR, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see "BLSUZZ2" on page 406)
2. Equate symbol record (see "BLSRESSY" on page 373)
3. PD, PDL
4. PXAO, PDE
5. PDAD, PDE for address range.

Output: The symbol record definition has been completed for this request. The 'X' symbol is then updated to indicate successful processing of the request.

BLSRADDt: Resolve Address PDEs

Function: This module supplies a data type when processing the address PDE expressions. If an explicit data type was specified, it calls module BLSRADDU.

Modules Called: BLSRADDU, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see "BLSUZZ2" on page 406)
2. Equate symbol record (see "BLSRESSY" on page 373)
3. PD, PDL
4. PXAO, PDE

Output: Data type information has been completed in the symbol record for this request.

BLSRADDU: Process Data Type

Function: This module processes the explicit data type information as supplied on the subcommand.

Modules Called: BLSUALLO, BLSUFRE1

1. Using the parse information, it indexes into a table of data type codes and completes the ESSYDTY field of the record.
2. It establishes a default length of 2 or 4 for signed data type.
3. If an explicit name is specified for area, module, or structure, it moves the name into the symbol record.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Equate symbol record (see “BLSRESSY” on page 373)
3. PD, PDL
4. PXAO, PDE

Output: Additional information is completed in the equate symbol record for the data type as described in the function above.

Data Areas Used: TYPTABLE (an array of supported data types).

BLSRADD1: Resolve Single Address Expression

Function: This module processes a single address PDE that is returned from the TSO parse routine. The module:

1. Processes symbolic names and relative addresses.
2. Processes register expressions (general and floating point) for both absolute and virtual storage.
3. Processes literal addresses.
4. Processes indirect address values.
5. Processes and computes address expression values.
6. Ensures that the resolved address is within the proper limits.

Modules Called: BLSRACC, BLSRADD1, BLSRADD2, BLSRADD3, BLSRADD4, BLSRDATA, BLSRESSA, BLSRESCK, BLSRESGA, BLSRMSGGA, BLSRSAGU, BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUPGME, BLSUSIGR, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Equate symbol record (see “BLSRESSY” on page 373)
3. PD, PDL
4. PXAO, PDE
5. PDAD, PDE for address range
6. PDLEN range indicator parameter

Value	Meaning
0	First and only address expression.
1	First address expression of range.
2	Second address expression of range.

Output: Additional information is completed in the equate symbol record for the functions described above.

BLSRADD2: Generate Address Space Information

Function: This module sets a default address space and calls BLSRADD3 to complete or override the address space description.

Modules Called: BLSRADD3, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Equate symbol record (see “BLSRESSY” on page 373)
3. PD, PDL
4. PXAO, PDE

Output: Address space information has been completed in the symbol record for this request.

BLSRADD3: Override Address Space Information

Function: BLSRADD3 processes the address space information.

1. It processes combinations of the absolute, CPU, header, ASID, and real, and status operands. It sets defaults for CPU address and ASID if necessary.
2. A specific CPU address is processed by calling BLSRADD4.

Modules Called: BLSRADD4, BLSRDUCK, BLSRESGU, BLSUALLO, BLSUFRE1

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Equate symbol record (see “BLSRESSY” on page 373)
3. PD, PDL
4. PXAO, PDE

Output: Address space information has been completed in the symbol record for this request.

BLSRADD4: Override CPU Address or ASID

Function: This module processes any explicit CPU or ASID specifications, and places that information in the symbol record.

Modules Called: BLSRESGU, BLSUALLO, BLSUFRE1

1. If CPU is explicitly specified, it saves the value in the symbol record.
2. If virtual storage and ASID are explicitly specified, it saves the ASID value in the symbol record.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Equate symbol record (see “BLSRESSY” on page 373)
3. PD, PDL
4. PXAO, PDE

Output: CPU and ASID fields are completed in the symbol record for this request.

BLSRADGP: Generate Extended Address Range PDE

Function: Process data described by an IKJIDENT PDE to generate an IKJPOSIT PDE for extended address notation. Storage acquired from subpool 1 for the IKJPOSIT PDE is added to the PDL defined by the caller to permit later release of the storage (IKJRLSA macro).

Modules Called: BLSUALLO, BLSUFRE1

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. An IKJIDENT PDE generated by IKJPARS
3. A fullword PDL address, or zero
4. Space for an “IKJPOSIT ADDRESS,RANGE,EXTENDED” PDE

Output

1. A fullword PDL address
2. An “IKJPOSIT ADDRESS,RANGE,EXTENDED” PDE

BLSRDATAK: Edit Storage Attributes

Function: Produce a storage description which is logically consistent.

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. An address space description
3. An address
4. An attribute description

Output

1. The address space description will be updated as required
2. The address will be updated as required
3. Attributes will be updated as required

BLSUSIGR: Resolve Signed Integer Range

Function: Translate data described by an IKJIDENT PDE generated by IKJPARS from EBCDIC to a signed binary integer range.

Modules Called: BLSUALLO, BLSUFRE1

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. An IKJIDENT PDE generated by IKJPARS
3. A fullword
4. A fullword

Output

1. A fullword signed binary integer defines the origin of the range
2. A fullword signed binary integer defines the end of the range

Dump Directory Handling

Dump directory handling subroutines fall in two categories:

- Most subroutines process a single VSAM request described by an RPL passed as a parameter and diagnose any problems encountered using the IKJEFF19 (VSAMFAIL) service routine.
- Other subroutines perform more complex activities associated with the dump directory.

Simple Request Handling

Simple request handlers all receive a common parameter list and have a similar processing sequence.

Modules Called: BLSUALLO, BLSUFRE1, IKJEFF19

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. RPL (macro IFGRPL)

Output: VSAM processing has been requested and performed. If an error condition was detected, IKJEFF19 has been used to diagnose the error.

Data Areas Used: RPL, ACB

Figure 122 provides more information regarding these routines.

Service Routine	VSAM Macro	Description
BLSUVSEN	ENDREQ	Terminate VSAM request string
BLSUVSRE	REASE	Erase a record
BLSUVSGE	GET	Get a record — record-not-found is acceptable
BLSUVSGU	GET	Get a record — no error is acceptable
BLSUVSPO	POINT	Reposition VSAM request string
BLSUVSPU	PUT	Put a record
BLSUVSWB	WRTBFR	Write updated control intervals

Figure 122. Dump Directory Simple Request Handling Summary

Complex Request Handling

The program control flow chart listing the modules used by the dump directory complex request handling function is presented in Figure 123 on page 311 .

BLSUVSAR - Add or replace records BLSUALLO - Allocate automatic storage BLSUVSGE - Get a record BLSUVSRE - Erase a record BLSUVSPU - Put a record BLSUVSEN - Terminate VSAM request string BLSUTRMO - Issue error messages BLSUFRE1 - Free automatic storage BLSUVSOP - Open initialization BLSUALLO - Allocate automatic storage BLSUTRMV - Issue error messages IKJEFF18 - Process dynamic allocation errors BLSUVSOQ - Open the dump directory IGGOCLA0 - SHOWCAT for dump directory attributes BLSUTRMO - Issue error messages IKJEFF19 - Diagnose VSAM errors BLSUFRE1 - Free automatic storage
--

Figure 123. Dump Directory Complex Request Control Flow

BLSUVSAR: Add or Replace Records in Dump Directory

Function: This module adds or replaces existing records in the dump directory data set.

1. It checks if the record already exists in the data set. If the record already exists and is exactly equal, then it exits. Otherwise, it deletes that record.
2. It puts the new input record in the data set by calling BLSUVSPU.

Modules Called: BLSUALLO, BLSUFRE1, BLSUTRMV, BLSUVSEN, BLSUVSRE, BLSUVSGE, BLSUVSMR, BLSUVSPU, BLSUVSWB

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. RPL (macro IFGRPL)

Output: The input record has been placed in the dump directory data set.

Data Areas Used: ACB

BLSUVSCA: Create an ACB

Function: Create an address method control block by means of a GENCB Macro-instruction. Diagnose any difficulty encountered.

Modules Called: BLSUALLO, BLSUFRE1, BLSUTRMV

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. A GENCB parameter list

Output: None

BLSUVSCL: Close the Dump Directory Data Set

Function: This module performs the CLOSE function for the dump directory. It:

1. Closes the dump directory data set.
2. Issues the FREEMAIN macro to release the BLUVSFB control block storage.
3. Deletes the VSAM resource pool.

Input: Master task variable (see “BLSUZZ2” on page 406)

Output: The dump directory has been closed and the resources returned to the system.

Data Areas Used: ACB

BLSUVSCR: Create a Request Parameter List (RPL)

Function: This module creates a request parameter list by means of a GENCB macro-instruction. It diagnoses any difficulty encountered.

Modules Called: BLSUALLO, BLSUFRE1, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A GENCB parameter list (macro GENCB), ready to be used to create a request parameter list (RPL)

Output: None

BLSUVSMR: Modify a Request Parameter List (RPL)

Function: This module modifies a request parameter list by means of a MODCB macro. It diagnoses any difficulty encountered.

Modules Called: BLSUALLO, BLSUFRE1, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. A MODCB parameter list (macro MODCB), ready to be used to modify an RPL.

Output: None

Data Areas Used: RPL

BLSUVSOP: Open Initialization for Dump Directory

Function: BLSUVSOP prepares for the open process of the dump directory by invoking dynamic allocation to ensure that DDNAME IPCSDDIR is allocated to a VSAM data set. If it is, BLSUVSOP calls BLSUVSOQ to open the dump directory.

Modules Called: BLSUALLO, BLSUFRE1, BLSUTRMV, BLSUVSOQ, IKJEFF18

Input: Master task variable (see “BLSUZZ2” on page 406)

Output: Either the dump directory has been opened successfully or an error message has been issued.

Data Areas Used: IEFZB4D0, IEFZB4D2

BLSUVSOQ: Open the Dump Directory Data Set

Function: This module performs the OPEN function on the dump directory. It:

1. Builds a VSAM local shared resource pool.
2. Issues the GETMAIN macro to obtain an area of storage for the BLSUVSFB (see “BLSUVSFB” on page 403).
3. Creates an ACB control block in ZZ4ACB using the GENCB process.
4. Prepares the ACB for I/O requests using the OPEN macro.
5. Assures the integrity of the dump directory using the VERIFY macro. This is only done when OPEN indicates a concern which may be corrected by VERIFY processing.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUTRMV, IKJEFF19

Input

1. Master task variable (see “BLSUZZ2” on page 406)
2. Data set name
3. The DDNAME

Output: Either the dump directory has been opened successfully or an error message has been issued.

Dump Directory Record Management

Each dump directory management routine receives a similar parameter list:

1. Active task variable (see “BLSUZZ2” on page 406).
2. A dump directory record. Some routines (BLSRxxGE) require that only key fields be completed.

The processing of a dump directory management routine may produce three results:

Routine	Result of Processing
BLSRxxAR	Merges the information supplied by the caller into the dump directory. Existing dump directory records may be deleted or updated to maintain logical consistency.
BLSRxxGx	Retrieves information requested by the caller from the dump directory.
BLSRxxPU	Stores new information supplied by the caller into the dump directory.

Figure 124 provides more information regarding these routines.

Module	Function	Modules Called
BLSRESAR	Add or replace the definition of a symbol in the dump directory.	BLSRESCK, BLSRRDAR, BLSUALLY, BLSUFRE1, BLSUPGME, BLSUVSAR, BLSUVSCR
BLSRESGC	Retrieve the definition of a symbol from the dump directory.	BLSUALLY, BLSUFRE1, BLSUVSCR, BLSUVSGE
BLSRESPU	Add the definition of a symbol to the dump directory.	BLSRESCK, BLSRRDAR, BLSUALLY, BLSUFRE1, BLSUPGME, BLSUVSCR, BLSUVSPU
BLSRPHAR	Add or replace a program history record in the dump directory.	BLSUALLY, BLSUFRE1, BLSUVSAR, BLSUVSCR
BLSRPHGE	Retrieve a program history record from the dump directory.	BLSUALLY, BLSUFRE1, BLSUVSCR, BLSUVSGE
BLSRRAAR	Add or replace address space information in the dump directory.	BLSUALLY, BLSUFRE1, BLSUVSCR, BLSUVSEN, BLSUVSER, BLSUVSGU, BLSUVSMR, BLSUVSPO, BLSUVSPU, IKJEFF19
BLSRRAGE	Retrieve address space information from the dump directory or from a translation process.	BLSRRACI, BLSRRACR, BLSRRACV, BLSRRBGE, BLSRRCGE, BLSUALLY, BLSUFRE1, BLSUVSCR, BLSUVSEN, BLSUVSGU
BLSRRBGE	Retrieve the location of a dump block from the dump directory.	BLSUALLY, BLSUFRE1, BLSUVSCR, BLSUVSEN, BLSUVSGU
BLSRRCGE	Verify the presence of a dump block using data stored in the dump directory.	BLSUALLY, BLSUFRE1, BLSUVSCR, BLSUVSEN, BLSUVSGU
BLSRRDAR	Add or replace a data set record in the dump directory.	BLSUALLY, BLSUFRE1, BLSUVSAR, BLSUVSCR, BLSUVSGU
BLSRRDGE	Retrieve a data set record from the dump directory.	BLSUALLY, BLSUFRE1, BLSUVSCR, BLSUVSGE
BLSRRDPU	Add a data set record to the dump directory. Associate the data set name with a reference number unique with the directory.	BLSRDUDR, BLSUALLY, BLSUFRE1, BLSUPGMS, BLSUVSCR, BLSUVSEN, BLSUVSGU, BLSUVSMR, BLSUVSPO, BLSUVSPU, BLSUVSWB
BLSRSAAR	Add or replace a storage address record in the dump directory.	BLSUALLY, BLSUFRE1, BLSUVSAR, BLSUVSCR
BLSRSAGC	Retrieve a storage address record from the dump directory.	BLSUALLY, BLSUFRE1, BLSUVSCR, BLSUVSGE
BLSRSAPU	Add a storage address record to the dump directory.	BLSUALLY, BLSUFRE1, BLSUVSCR, BLSUVSPU
BLSRSDAR	Add or replace a session defaults record in the dump directory.	BLSUALLY, BLSUFRE1, BLSUVSAR, BLSUVSCR
BLSRSDGE	Retrieve a session defaults record from the dump directory.	BLSUALLY, BLSUFRE1, BLSUVSCR, BLSUVSGE

Figure 124. Dump Directory Record Management Summary

Dump Data Formatting

Dump data formatting modules may usefully be viewed as three groups:

- Basic formatters
- Storage formatters
- Support modules

Basic Formatters

Basic formatters all receive a common parameter list and all generate a display, returning only a return code to the caller to indicate whether the display was transmitted successfully.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Equate symbol record parameter

Figure 125 supplies more information regarding basic formatters. Some of these modules have multiple entry point. When this is the case, the primary entry point is listed first, and the secondary entry points are alphabetized below the primary entry point.

Module	Function	Modules Called
BLST01	Display REQUEST, SYMBOL, REMARK, MACHINE, and/or STORAGE.	BLSRESCK, BLST03, BLST08
BLST03	Display REQUEST, SYMBOL, and/or REMARK	BLSUPUTA, BLSRMSG, BLSRMSG, BLSRMSGD, BLSUPUTC
BLST04	Display MACHINE	BLST04A, BLST04BL, BLST04BS, BLST04C, BLST04CR, BLST04CV, BLST04H
BLST04A	Display MACHINE — ABSOLUTE	BLSUMPK1, BLSUPUTV
BLST04BS BLST04BL BLST04C BLST04H	Display MACHINE — RBA Display MACHINE — BLOCK Display MACHINE — CPU STATUS Display MACHINE — HEADER	BLSRMSG, BLSUMPK1, BLSUPUTO
BLST04CV BLST04CR	Display MACHINE — ASID Display MACHINE — CPU REAL	BLSRMSG, BLSUMPK1, BLSUPUTO
BLST08	Display MACHINE and/or STORAGE	BLSRACCQ, BLST06, BLSRRAGE, BLSUMPK1, BLSUPUTA

Figure 125. Summary of Basic Formatters

Storage Formatters

Storage formatters all receive a common parameter list and all generate a display of dump data, returning only a return code to the caller to indicate whether the display was transmitted successfully.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Equate symbol record parameter
3. Input data buffer

Figure 126 supplies more information regarding storage formatters. Some of these modules have multiple entry point. When this is the case, the primary entry point is listed first, and the secondary entry points are alphabetized below the primary entry point.

Module	Function	Modules Called
BLSTB BLSTA	Display STORAGE — BIT Display STORAGE — POINTER	BLSTB4, BLSTB5
BLSTB4	Display 16 bytes of bit string data per line	BLSUMPK1, BLSUPUTA
BLSTB5	Display 32 bytes of bit string data per line	BLSUMPK1, BLSUPUTA
BLSTC	Display STORAGE — CHARACTER	BLSUMPK1, BLSUPUTA
BLSTF	Display STORAGE — SIGNED	BLST05, BLSTB
BLSTM BLSTL	Display STORAGE — STRUCTURE Display STORAGE — MODULE	BLSTM4, BLSTM5
BLSTM4	Display 16 bytes of structured data per line	BLSUMPK1, BLSUPUTA
BLSTM5	Display 32 bytes of structured data per line	BLSUMPK1, BLSUPUTA
BLSTR	Display registers	BLSUPUTA, BLSTB
BLSTU	Display STORAGE — AREA	BLSTU4, BLSTU5
BLSTU4	Display 16 bytes of area data per line	BLSUMPK1, BLSUPUTA
BLSTU5	Display 32 bytes of area data per line	BLSUMPK1, BLSUPUTA
BLSTY	Display STORAGE — UNSIGNED	BLST05, BLSTB
BLST02	Display REQUEST, SYMBOL, REMARK, MACHINE, and/or STORAGE.	BLSRESCK, BLST03, BLST06
BLST06	Display data	BLSTA, BLSTB, BLSTC, BLSTF, BLSTL, BLSTM, BLSTR, BLSTU, BLSTY

Figure 126. Summary of Storage Formatters

Support Modules

BLSRECHA: Format Character Data

Function: Format character data.

Modules Called: BLSUALLO, BLSUFRE1

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. V1 (VRC)
3. P (PRM)
4. PKEDBLNK (KEYWD)
5. PKEDQUOT (KEYWD)

Output

1. V1 (VRC) contains the formatted result
2. P (PRM) is destroyed

BLSRMDSN: Format Dump Source Name

Function: Format a keyword or keyword and value describing the current dump source.

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. Message buffer

Output: The dump source name is in the message buffer.

BLSRMSGGA: Address Space Name Formatter

Function: BLSRMSGGA builds address space name information in a supplied input buffer. This is for virtual storage, real storage, absolute storage, CPU status, and storage qualified by a CPU address. The buffer is built in text insertion format as follows:

1. The length field is set to the length of the buffer.
2. The offset field is zeroed.
3. The first character of the text field is always blank.
4. The remaining characters contain the address space description.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPK1

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Message buffer
3. RDAI, address space identification

Output: Address space information has been built in the input message buffer.

BLSRMSGGB: Storage Description Formatter

Function: BLSRMSGGB builds a storage description in a supplied input buffer. The description includes length and entry information. The text insertion buffer is built as follows:

1. The length field is set to the length of the buffer.
2. The offset field is zeroed.
3. The first character of the text field is always blank.
4. The remaining characters of text contain the storage description.

Modules Called BLSUALLO, BLSUFRE1, BLSUMPK1

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Message buffer
3. DPA, storage description block

Output: Address space information has been built in the input message buffer.

BLSRMSGD: Data Type Formatter

Function: BLSRMSGD builds the data type description in a supplied input buffer. The data type description includes pointer, bit, character, signed, module, unsigned, structure, and area. The text insertion buffer is built as follows:

1. The length field is set to the length of the buffer.
2. The offset field is zeroed.
3. The first character of the text is always blank.
4. The remaining text contains the data type description.

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Message buffer
3. DTYP, data type description

Output: Address space information has been built in the input message buffer.

BLSRM154: Transmit BLS18154I

Function: BLSRM154 formats and transmits message BLS18154I.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Type of dump
3. Phase of processing

Output: None

BLSRM185: Transmit BLS18185I

Function: BLSRM185 formats and transmits message BLS18185I.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Type of dump

Output: None

BLSRM192: Transmit BLS18192I

Function: BLSRM192 formats and transmits message BLS18192I.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Maximum number of digits
3. Type of value

Output: None

BLSRM193: Transmit BLS18193I

Function: BLSRM193 formats and transmits message BLS18193I.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Type of value

Output: None

BLSRM199: Transmit BLS18199I

Function: BLSRM199 formats and transmits message BLS18199I.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Maximum value supported
3. Type of value

Output: None

BLSRM200: Transmit BLS18200I

Function: BLSRM200 formats and transmits message BLS18200I.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Minimum value supported
3. Type of value

Output: None

BLSRM214: Transmit BLS18214I

Function: BLSRM214 formats and transmits message BLS18214I.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Type of string.

Output: None

BLSRM300: Transmit BLS18300I

Function: BLSRM300 transmits message BLS18300I.

Modules Called: BLSUALLO, BLSUFRE1, BLSUPUTV

Input: Active task variable (see “BLSUZZ2” on page 406)

Output: None

BLSRM301: Transmit BLS18301I

Function: BLSRM301 formats and transmits message BLS18301I.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Alignment required

Output: None

BLSRM302: Transmit BLS18302I

Function: BLSRM302 formats and transmits message BLS18302I.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPKN, BLSUTRMV

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Locator descriptor which supplies:
 - The number of locators
 - The length of each locator name
 - The names of each locator
3. Error flag array (1 byte per locator)
4. Locator value array (1 word per locator)

Output: None

***BLSRZALC*: Translate Binary To Modulo 26**

Function: Translate a binary integer to an EBCDIC Mod26 suffix.

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. Suffix buffer
3. Unsigned binary fullword integer

Output

1. The active task variable (see “BLSUZZ2” on page 406)
2. Suffix buffer

***BLSTF2, BLSTF4*: Format Signed Halfword or Fullword Data**

Function: Format halfword (BLSTF2) or fullword (BLSTF4) signed binary data.

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. JH data to be formatted
3. JS initial input column
4. JI number of signed binary data items
5. M2 output data buffer
6. VAL1 first formatted value

Output: The value is formatted and returned

***BLSTY1, BLSTY2, BLSTY3, BLSTY4*: Format 1-Byte, 2-Byte, 3-Byte, or 4-Byte Unsigned Data**

The modules listed below perform the functions of formatting the input data from unsigned binary to EBCDIC, and returning the results to the caller.

Module	Input Format
BLSTY1	1 byte unsigned binary
BLSTY2	2 byte unsigned binary
BLSTY3	3 byte unsigned binary
BLSTY4	4 byte unsigned binary

Input

1. Active task variable (see “BLSUZZ2” on page 406)
2. Data to be formatted
3. Initial input column
4. Number of data items
5. Output data buffer
6. First formatted value

Output: The input data is formatted

BLSTY4C3, BLSTY4C4, BLSTY8C1, BLSTY8C2: Compare 1-Byte, 2-Byte, 3-Byte, or 4-Byte Data

The modules listed below determine whether all fields in an array are equal.

Module	Array Structure
BLSTY4C3	Four 3-byte fields
BLSTY4C4	Four 4-byte fields
BLSTY8C1	Eight 1-byte fields
BLSTY8C2	Eight 2-byte fields

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. An array of data

Output: None

BLST05: Display Binary Data

Function: BLST05 is called by the binary signed and unsigned routines to display storage. The addresses of the formatting and repetitive line checking subroutines are passed in as parameters from the BLSTF (signed) and BLSTY (unsigned) routines.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUPUTA

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. Equate symbol record
3. Input data buffer
4. Number of items per line
5. Indentation control 1
6. Indentation control 2
7. Routine to fill buffer
8. Routine to check for duplicates

Output: None

BLSUMDSN: Format Data Set Name

Function: Build the data set name in the supplied buffer.

Input

1. The active task variable (see “BLSUZZ2” on page 406)
2. A message buffer

Output: The requested data is placed in the buffer

BLSUMPKN: Compress a Message

Function: BLSUMPKN removes extraneous blanks from the text of a message. It ignores a specified number of characters of message text because the caller knows that they contain no extraneous blanks.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. A variable-length message buffer. (The first halfword of a variable-length message indicates the length of the message.)
3. The address of a fullword binary integer specifying the number of bytes of text to be exempt from compression.

Output: The first halfword is updated to reflect the length of the edited message, and all extraneous blanks are removed.

BLSUMPK1: Compress a Message

Function: BLSUMPK1 removes extraneous blanks from the text of a message. It ignores the first character of message text.

Modules Called: BLSUALLO, BLSUFRE1, BLSUMPKN

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. A variable-length message buffer. (The first halfword of a variable-length message indicates the length of the message.)

Output: The first halfword is updated to reflect the length of the edited message, and all extraneous blanks are removed.

BLSUMTOD: Format TOD Clock Value

Function: BLSUMTOD produces a buffer containing the date and time in standard MVS/IPCS format, mm/dd/yy hh:mm:ss, given an 8-byte time-of-day clock value.

Input

1. Time-of-day clock value
2. 17-byte buffer

Output: The buffer contains a representation of the value in the format mm/dd/yy hh:mm:ss.

BLS9MPKN: Compress a message

Function: Remove extraneous blanks from the text of a message. Ignore a specified number of characters of message text because the caller knows that they contain no extraneous blanks.

Input

1. A variable-length message buffer
2. The address of a fullword integer specifying the number of bytes to be exempt from compression

Output: The first halfword will be updated to the new message length, and blanks are removed from the message

Inter-Task Request Servers

All inter-task request servers receive control under the IPCS master task. They receive the same parameter list and produce comparable output.

Input

1. Active task variable (see “BLSUZZ2” on page 406).
2. The IPCS task variable (see “BLSUZZ2” on page 406) associated with the requesting task. Field ZZ2ITRPP must contain the address of the parameter list for the server.

Output: None

See Figure 127 for more information regarding these routines.

Module	Service	Modules Called
BLSUIT01	DYNALOC (SVC 99)	BLSUALLO, BLSUDYNA, BLSUFRE1
BLSUIT02	OPEN (SVC 19)	BLSUALLO, BLSUFRE1
BLSUIT03	CLOSE (SVC 20)	BLSUALLO, BLSUFRE1
BLSUIT04	OPEN print file	BLSUALLO, BLSUFRE1, BLSUMPK1, BLSUPRAB, BLSUPRDX, BLSUTRMV
BLSUIT05	CLOSE print file	BLSUALLO, BLSUFRE1, BLSUTRMV

Figure 127. Inter-Task Service Summary

Section 4. Data Areas

This section briefly describes the IPCS data areas. An overview of the IPCS data structure is presented in “Data Structure” on page 4.

Each description consists of three parts:

1. An acronym that is commonly associated with the data area.
2. A description of the characteristics of the data area. This discussion can vary but always contains the following information:
 - a. The “Common Name” of the control block briefly highlights an important characteristic of the data area, one which distinguishes it from all other IPCS data areas.
 - b. The “Macro ID” identifies the IPCS macro used to describe the contents of the data area.
 - c. The “Function” paragraph provides a more detailed description of the data area and the usage it receives.
3. A description of the detailed structure of the data area. Each field is named and its function is described. Two kinds of fields appear:
 - a. Fields that occupy an integral number of bytes are characterized by a decimal offset, a hexadecimal offset (in parentheses), an indication of the type of data contained in the field, and a length expressed in bytes.
 - b. Fields that occupy part of a byte (usually individual flag bits) are characterized by a mask in which all bits in the byte are shown. Each bit that is part of the field is indicated by the digit “1.” Each bit that is not part of the field is indicated by a period.

The following data areas are described:

“BLSABDPL” on page 326	“BLSRFD” on page 376
“BLSBVT” on page 333	“BLSRHSD” on page 378
“BLSDMCB” on page 336	“BLSRPHSY” on page 381
“BLSDSD” on page 342	“BLSRPRX” on page 382
“BLSDVT” on page 343	“BLSRRASY” on page 385
“BLSFP” on page 345	“BLSRRBSY” on page 387
“BLSLBLS” on page 346	“BLSRRCSY” on page 388
“BLSLNTRC” on page 352	“BLSRRDSY” on page 389
“BLSPDR” on page 357	“BLSRSASY” on page 391
“BLSQEXTP” on page 360	“BLSRSDSY” on page 394
“BLSQSMPL” on page 361	“BLSRSST” on page 395
“BLSQSRA” on page 363	“BLSRVALY” on page 396
“BLSRDATC” on page 365	“BLSRVT” on page 397
“BLSRDATA” on page 367	“BLSRZZ6” on page 400
“BLSRDATT” on page 369	“BLSUVSFB” on page 403
“BLSRDTDT” on page 370	“BLSUZZ1” on page 404
“BLSRDUT” on page 371	“BLSUZZ2” on page 406
“BLSRESSY” on page 373	

BLSABDPL

Common Name: Common Exit Parameter List

Macro ID: BLSABDPL

Created by: PRDMP, IPCS, and SNAP

Pointed to by: Register 1 upon entry to a dump processing exit.

Function: Provide the addresses of subroutines and data to dump processing exits by the MVS/XA dump processing host programs:

- PRDMP
- IPCS
- SNAP

The BLSABDPL interface allows dump processing exits to pass information to requested subroutines.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	96	ABDPL	Common AMDPRDMP/IPCS/SNAP parameter list to exits
0	(0) ADDRESS	4	ADPLTCB	TCB of task being displayed
4	(4) SIGNED	2	ADPLASID	Address space identifier
6	(6) CHARACTER	1	ADPLSBPL	Subpool used to get save area by component routine
7	(7) CHARACTER	1	ADPLFLAG	Flag field
1... ..			ADPLSNPR	0-Module loaded by SNAP 1-Module loaded by PRDMP/IPCS
.1... ..			ADPLSYTM	0-System is OS/VS2 1-System is OS/VS1
..1.			ADPLDMGT	For data management formatter under SNAP: 0-Format DEB only, 1-Format DEB,DCB,I0B
...1			ADPLIPCS	Called by IPCS
.... 1...			ADPLPRT	SYSPRINT data set
.... .1..			ADPLSYNO	Exit given control for syntax checking only
.... ..1.			ADPLEJEC	Page eject
8	(8) ADDRESS	4	ADPLBUF	Pointer to output buffer
12	(C) ADDRESS	4	ADPLPRNT	Address of print routine
16	(10) ADDRESS	4	ADPLCVT	Address of CVT
20	(14) ADDRESS	4	ADPLMEMA	Address of memory access routine
24	(18) ADDRESS	4	ADPLFRMT	Address of format routine
28	(1C) SIGNED	4	ADPLCOM1	Reserved for component use
32	(20) SIGNED	4	ADPLCOM2	Reserved for component use
36	(24) SIGNED	4	ADPLCOM3	Reserved for component use
40	(28) SIGNED	4	ADPLCOM4	Reserved for component use
44	(2C) ADDRESS	4	ADPLFMT1	Reserved for format routine
48	(30) ADDRESS	4	ADPLFMT2	Reserved for format routine
52	(34) ADDRESS	4	ADPLEXT	Address of extension whose first word contains the address of the operands list or zero if none

BLSABDPL - Storage Layout (continued)

Offsets	Type	Length	Name	Description
56	(38) ADDRESS	4	ADPLABDA	Address of host internal parameter list
60	(3C) ADDRESS	4	ADPLTRFM	Address of trace control block (SNAP only)
64	(40) ADDRESS	4	ADPLSERV	->Services router
68	(44) UNSIGNED	1	ADPLLEV	Index indentation level
69	(45) UNSIGNED	1	ADPLIDX	Entry code number corresponding to AMDMNDXT macro entries
70	(46) UNSIGNED	2	ADPLLNCT	Lines per page
72	(48) UNSIGNED	2	ADPLLNRM	Lines remaining on the current page
74	(4A) UNSIGNED	2	ADPLDLEN	Storage access length
76	(4C) UNSIGNED	2	ADPLOPLN	Length of operands
78	(4E) CHARACTER	1	ADPLPRDP	Address qualification
	1... ..		ADPLVIRT	Virtual address
	.1..		ADPLREAL	Real address
	..1.		ADPLCPU	CPU data request
	...1		ADPLHDR	Dump header request
	 111.			reserved

The following bit governs the masking of register zero prior to its use by the storage access service as a virtual storage address. If it is off, X'7FFFFFFF' will be logically ANDed with register zero to obtain the requested address. If it is on, X'00FFFFFF' will be logically ANDed with register zero to obtain the requested address.

79	 1 (4F) CHARACTER	1	ADPLSAMK	MVS/370 virtual address reserved
80	(50) ADDRESS	4	ADPLNDX	Address of the AMDPRNDX routine. Build index work entries
84	(54) UNSIGNED	4	ADPLPGNO	Current page number
88	(58) ADDRESS	4	ADPLSRA	->Services Router area
92	(5C) CHARACTER	4		reserved
0	(0) STRUCTURE	64	ADPLEXTN	ABDPL extension

ABDPL extension is pointed to by ADPLEXT.

0	(0) ADDRESS	4	ADPLOPTR	->Operands buffer
4	(4) ADDRESS	4	ADPLCPPL	->TSO CPPL
8	(8) ADDRESS	4	ADPLRSV	reserved
12	(C) UNSIGNED	2	ADPLCODE	Router return code
14	(E) BITSTRING	1	ADPLPFLG	Processing flags
	1... ..		ADPLNMSG	1-Suppress error messages
	.111 1111			reserved
15	(F) BITSTRING	1	ADPLEFLG	Error flags
	1... ..		ADPLEFAS	No automatic storage
	.111 1111			reserved

BLSABDPL - Storage Layout (continued)

Offsets	Type	Length	Name	Description
16	(10) UNSIGNED	1	ADPLMAXL	Recommended maximum line width
17	(11) UNSIGNED	1	ADPLSCOL	Control block formatting start column
18	(12) UNSIGNED	1	ADPLCOLS	Control block formatting column spacing
19	(13) CHARACTER	45	ADPLRSV1	reserved
0	(0) STRUCTURE	64	ADPLPACC	Storage access parameter list

Storage access service parameter list

0	(0) ADDRESS	4	ADPLPAAD	Address to be read
4	(4) ADDRESS	4	ADPLPART	Buffer location of data
8	(8) CHARACTER	56		reserved
0	(0) STRUCTURE	64	ADPLPECT	ECT exit parameter list

ECT service parameter list

0	(0) BITSTRING	1	ADPLPEFG	Flags for choosing exit '0' indicates verb exit TCB exit ASCB exit FORMAT exit PRINT CURRENT/JOBNAMES exit PRINT NUCLEUS exit Header exit Reserved
	1... ..		ADPLPETB	
	.1... ..		ADPLPEAS	
	.1... ..		ADPLPEFM	
	...1 ..		ADPLPEPJ	
	 1...		ADPLPEPN	
	1..		ADPLPEHD	
	11			Reserved
1	(1) BITSTRING	1	ADPLPERR	Flags for returning information to caller
	1... ..		ADPLPEST	Exit return code='4' indicating insufficient storage
	.1... ..		ADPLPENV	Verb not found in ECT
	.1... ..		ADPLPELI	LINK failed 806 abend
	...1 ..		ADPLPENE	No ESTAE established
	 1111			Reserved
2	(2) CHARACTER	2		Reserved
4	(4) CHARACTER	8	ADPLPEVB	VERB to search on
12	(C) CHARACTER	52		Reserved
0	(0) STRUCTURE	64	ADPLPFMT	Common parameter list

Control block and Model formatter services parameter list

0	(0) BITSTRING	1	ADPLPOPT	Option flags
	1... ..		ADPLPOAC	Check acronym
	.1... ..		ADPLPOIX	Index entry wanted
	.1... ..		ADPLPOLM	Line mode
	...1 ..		ADPLPOCL	Final host invocation to delete modules and freemain CBAT Load Table
	 1...		ADPLPSOF	Suppress offsets
	1..		ADPLPPDA	Print dump address
	1..		ADPLPSDH	Suppress dump header
	1		ADPLPSTM	Suppress truncation msg
1	(1) BITSTRING	2	ADPLPRET	Flags for returning information to caller

BLSABDPL - Storage Layout (continued)

Offsets	Type	Length	Name	Description
1	(1) BITSTRING	1	ADPLPRE1	Control block acronym check failed Unable to load model Unavailable control block Unable to format Truncated block No CBAT entry Null CBAT entry No CBAT load table storage
	1... ..		ADPLPRAC	
	.1... ..		ADPLPRNL	
	..1.		ADPLPRNB	
	...1		ADPLPRNF	
	 1...		ADPLPRTB	
	1..		ADPLPRNC	
	1.		ADPLPRNE	
	1		ADPLPRNG	
2	(2) BITSTRING	1	ADPLPRE2	
	1... ..		ADPLPRU	Formatter previously marked unuseable
	.1... ..		ADPLPRIM	Invalid model
	..1.		ADPLPRCM	Model error
	...1		ADPLPNVM	No view match, no output
	 1111			Reserved
3	(3) UNSIGNED	1	ADPLPBLC	Blank line count
4	(4) CHARACTER	8	ADPLPCHA	Control block acronym or model name
12	(C) ADDRESS	4	ADPLPPTR	Model address
16	(10) ADDRESS	4	ADPLPBAS	Buffer address if block is in main storage
20	(14) SIGNED	2	ADPLPDAC	Dynamic array count
22	(16) SIGNED	2	ADPLPBLS	Length of block in main storage
24	(18) ADDRESS	4	ADPLPBAV	Block's virtual address in dump
28	(1C) ADDRESS	4	ADPLPLME	Line mode exit entry point address
32	(20) BITSTRING	2	ADPLPVCL	Field viewing control
32	(20) BITSTRING	1	ADPLPVC1	Keyfield Summary field Register save area Linkage field Error fields Hex dump All non reserved fields Reserved fields
	1... ..		ADPLPKEY	
	.1... ..		ADPLPSUM	
	..1.		ADPLPREG	
	...1		ADPLPLIN	
	 1..		ADPLPEFD	
	1..		ADPLPHEX	
	1.		ADPLPNOR	
	1		ADPLPRES	
33	(21) BITSTRING	1	ADPLPVC2	
	1... ..		ADPLPSTA	Static array
	.1... ..		ADPLPDYN	Dynamic array
	..11			Reserved
	 1...		ADPLPCV1	Component use
	1..		ADPLPCV2	Component use
	1.		ADPLPCV3	Component use
	1		ADPLPCV4	Component use
34	(22) SIGNED	2	ADPLPOSI	Starting offset
36	(24) UNSIGNED	4	ADPLPDL1	Dimension 1 lower limit
40	(28) UNSIGNED	4	ADPLPDL2	Dimension 2 lower limit
44	(2C) UNSIGNED	4	ADPLPDU1	Dimension 1 upper limit
48	(30) UNSIGNED	4	ADPLPDU2	Dimension 2 upper limit
52	(34) CHARACTER	12		Reserved
0	(0) STRUCTURE	64	ADPLPSEL	Select ASID parameter list

Select ASID service parameter list

BLSABDPL - Storage Layout (continued)

Offsets	Type	Length	Name	Description
0	(0) BITSTRING 1... .. .1..1.1 1...1..11	1	ADPLPSF1 ADPLPSAL ADPLPSCR ADPLPSE ADPLPSTE ADPLPSJL ADPLPSAS	Keyword flags ALL CURRENT ERROR TCBERROR JOBLIST ASIDLIST Reserved
1	(1) CHARACTER	3		Reserved
4	(4) ADDRESS	4	ADPLPSOL	->Select service output list
8	(8) ADDRESS	4	ADPLPSJN	->Jobname list
12	(C) ADDRESS	4	ADPLPSAI	->ASID list
16	(10) CHARACTER	48		reserved
0	(0) STRUCTURE	8	ADPLOSEL	Select ASID output list

Select Service output mapping.

0	(0) CHARACTER	8	ADPLOS HD	Header
0	(0) SIGNED	2	ADPLOSSZ	Size of entire list
2	(2) SIGNED	2	ADPLOSCT	Count of entries
4	(4) BITSTRING 1... .. .1..1.1 1111	1	ADPLOSSF ADPLOSJM ADPLOS AU ADPLOS AM	Selection flags Some jobname(s) not found Some ASID(s) unassigned Some ASID(s) not in dump Reserved
5	(5) BITSTRING 1... .. .111 1111	1	ADPLOSDF ADPLOSSD	Dump flags Standalone dump Reserved
6	(6) CHARACTER	1		Reserved
7	(7) CHARACTER	1	ADPLOSSP	Subpool
8	(8) CHARACTER	0	ADPLOSNT	Array, one per selected ASID
8	(8) ADDRESS	4	ADPLOSAP	Pointer to ASCB
12	(C) CHARACTER	8	ADPLOS CW	Control word
12	(C) BITSTRING 1... .. .1..1.1 1...1..11	1	ADPLOS F1 ADPLOS CR ADPLOS JN ADPLOS ER ADPLOS TE ADPLOS AS ADPLOS AL	Flag for selection match Current on CPU=ASLCPUX Matches jobnames(ASLJOBX) ASID in error TCBERROR Selected by ASIDLIST Selected by ALL Reserved
13	(D) BITSTRING 1... .. .1..1.1 1...1..11	1	ADPLOS F2 ADPLOS HA ADPLOS PA ADPLOS SA ADPLOS CM ADPLOS ND ADPLOS NA	Status flags CURRENT HOME ASID CURRENT PRIMARY ASID CURRENT SECONDARY ASID ASID HOLDS CML LOCK Address space not in dump Private area not accessed Reserved
14	(E) UNSIGNED	1	ADPLOS CP	CPUID where current
15	(F) CHARACTER	5		Reserved

BLSABDPL - Storage Layout (continued)

Offsets	Type	Length	Name	Description	
20	(14)	SIGNED	2	ADPLOSAI	ASID
22	(16)	CHARACTER	2		Reserved
24	(18)	CHARACTER	8	ADPLOSJB	Jobname
32	(20)	CHARACTER	8		RESERVED
0	(0)	STRUCTURE	192	AMDCPMAP	Status record data

AMDCPMAP is a mapping of the CPU STATUS data obtained from a dump.

0	(0) BITSTRING 1... .. .1...1.1	1	AMDCFLAG AMDCUNIP AMDCSINV AMDCGPRV AMDCSADP	CPU status flags CPU is a uniprocessor. Processor address is invalid. Stand alone dump unable to perform store status. Only processor address is valid. Operator did not perform store status. Only general registers and, if MP, pro- cessor address are valid. Not from a stand alone dump
1	(1) CHARACTER	1	AMDCPAD1	Padding
2	(2) UNSIGNED	2	AMDCPADR	Processor address
4	(4) CHARACTER	168	AMDCREGS	REGs and current PSW
4	(4) CHARACTER	32	AMDCFREG	Floating point REGs
36	(24) CHARACTER	64	AMDCGREG	General registers
100	(64) CHARACTER	64	AMDCCREG	Control registers
164	(A4) CHARACTER	8	AMDCCPSW	Current PSW
172	(AC) ADDRESS	4	AMDCPREG	Prefix register
176	(B0) CHARACTER	8	AMDCTIME	CPU timer value
184	(B8) CHARACTER	8	AMDCLOCK	Clock comparator

Alphabetic Index to Field Names

Name	Offset		Name	Offset		Name		
ABDPL	0		ADPLABDA	38		ADPLASID	4	
ADPLBUF	8		ADPLCODE	C		ADPLCOLS	12	
ADPLCOM1	1C		ADPLCOM2	20		ADPLCOM3	24	
ADPLCOM4	28		ADPLCPPL	4		ADPLCPU	4E	20
ADPLCVT	10		ADPLDLEN	4A		ADPLDMGT	7	20
ADPLEFAS	F	80	ADPLEFLG	F		ADPLEJEC	7	02
ADPLEXT	34		ADPLEXTN	0		ADPLFLAG	7	
ADPLFMT1	2C		ADPLFMT2	30		ADPLFRMT	18	
ADPLHDR	4E	10	ADPLIDX	45		ADPLIPCS	7	10
ADPLLEV	44		ADPLLNCT	46		ADPLLNRM	48	
ADPLMAXL	10		ADPLMEMA	14		ADPLNDX	50	
ADPLNMSG	E	80	ADPLOPLN	4C		ADPLOPTR	0	
ADPLOSAL	14		ADPLOSAL	C	04	ADPLOSAM	4	20
ADPLOSAP	8		ADPLOSAS	C	08	ADPLOSAU	4	40
ADPLOSCH	D	10	ADPLOSCP	E		ADPLOSCR	C	80
ADPLOSCT	2		ADPLOSCW	C		ADPLOSDF	5	
ADPLOSEL	0		ADPLOSER	C	20	ADPLOSFI	C	
ADPLOSFB	D		ADPLOSHA	D	80	ADPLOSHD	0	
ADPLOSJB	18		ADPLOSJM	4	80	ADPLOSJN	C	40

BLSABDPL - Alphabetic Index to Field Names (continued)

Name	Offset		Name	Offset		Name		
ADPLOSNA	D	04	ADPLOSND	D	08	ADPLOSNT	8	
ADPLOSSPA	D	40	ADPLOSSA	D	20	ADPLOSSD	5	80
ADPLOSSF	4		ADPLOSSP	7		ADPLOSSZ	0	
ADPLOSST	C	10	ADPLPAAD	0		ADPLPACC	0	
ADPLPART	4		ADPLPBAS	10		ADPLPBAV	18	
ADPLPBL	3		ADPLPBLS	16		ADPLPCHA	4	
ADPLPCV1	21	08	ADPLPCV2	21	04	ADPLPCV3	21	02
ADPLPCV4	21	01	ADPLPDAC	14		ADPLPDL1	24	
ADPLPDL2	28		ADPLPDU1	2C		ADPLPDU2	30	
ADPLPDYN	21	40	ADPLPEAS	0	40	ADPLPECT	0	
ADPLPEFD	20	08	ADPLPEFG	0		ADPLPEFM	0	20
ADPLPEHD	0	04	ADPLPELI	1	20	ADPLPENE	1	10
ADPLPENV	1	40	ADPLPEPJ	0	10	ADPLPEPN	0	08
ADPLPERR	1		ADPLPEST	1	80	ADPLPETB	0	80
ADPLPEVB	4		ADPLPFLG	E		ADPLPFMT	0	
ADPLPGNO	54		ADPLPHEX	20	04	ADPLPKEY	20	80
ADPLPLIN	20	10	ADPLPLME	1C		ADPLPNOR	20	02
ADPLPNVM	2	10	ADPLPOAC	0	80	ADPLPOCL	0	10
ADPLPOIX	0	40	ADPLPOLM	0	20	ADPLPOPT	0	
ADPLPOSI	22		ADPLPPDA	0	04	ADPLPPTR	C	
ADPLPRAC	1	80	ADPLPRCM	2	20	ADPLPRDP	4E	
ADPLPREG	20	20	ADPLPRES	20	01	ADPLPRET	1	
ADPLPRE1	1		ADPLPRE2	2		ADPLPRIM	2	40
ADPLPRNB	1	20	ADPLPRNC	1	04	ADPLPRNE	1	02
ADPLPRNF	1	10	ADPLPRNG	1	01	ADPLPRNL	1	40
ADPLPRNT	C		ADPLPRT	7	08	ADPLPRTB	1	08
ADPLPRUU	2	80	ADPLPSAI	C		ADPLPSAL	0	80
ADPLPSAS	0	04	ADPLPSCR	0	40	ADPLPSDH	0	02
ADPLPSEL	0		ADPLPSER	0	20	ADPLPSF1	0	
ADPLPSJL	0	08	ADPLPSJN	8		ADPLPSOF	0	08
ADPLPSOL	4		ADPLPSTA	21	80	ADPLPSTE	0	10
ADPLPSTM	0	01	ADPLPSUM	20	40	ADPLPVCL	20	
ADPLPVC1	20		ADPLPVC2	21		ADPLREAL	4E	40
ADPLRSV	8		ADPLRSV1	13		ADPLSAMK	4E	01
ADPLSBPL	6		ADPLSCOL	11		ADPLSERV	40	
ADPLSNPR	7	80	ADPLSRA	58		ADPLSYNO	7	04
ADPLSYTM	7	40	ADPLTCB	0		ADPLTRFM	3C	
ADPLVIRT	4E	80	AMDCCPSW	A4		AMDCCREG	64	
AMDCFLAG	0		AMDCFREG	4		AMDCGPRV	0	20
AMDCGREG	24		AMDCLOCK	B8		AMDCPADR	2	
AMDCPAD1	1		AMDCPMAP	0		AMDCPREG	AC	
AMDCREGS	4		AMDCSADP	0	10	AMDCSINV	0	40
AMDCTIME	R0		AMDCUNIP	0	80			

BLSBVT

Common Name: Common Data Vector

Macro ID: BLSUSERV

Created by: Compilation of module BLSUVECT

Pointed to by: Field ZZ2BVTP (see “BLSUZZ2” on page 406)

Function: Provide the addresses of subroutines and data shared by all subcomponents of IPCS.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	384	BVT	BLRVCT #MD83078
0	(0) ADDRESS	12	BVT0001P	reserved
12	(C) ADDRESS	4	BVTPUTOP	->BLSUPUTO
16	(10) ADDRESS	4	BVT0005P	reserved
20	(14) ADDRESS	4	BVTPUTVP	->BLSUPUTV
24	(18) ADDRESS	4	BVTPUTAP	->BLSUPUTA
28	(1C) ADDRESS	8	BVT0008P	reserved
36	(24) ADDRESS	4	BVTPUTDP	->BLSUPUTD
40	(28) ADDRESS	4	BVT0011P	reserved
44	(2C) ADDRESS	4	BVTMON2P	->BLSUMON2
48	(30) ADDRESS	4	BTVSARP	->BLSUVSAR
52	(34) ADDRESS	12	BVT0014P	reserved
64	(40) ADDRESS	4	BTVSENP	->BLSUVSEN
68	(44) ADDRESS	4	BTVSERP	->BLSUVSER
72	(48) ADDRESS	4	BTVSGEP	->BLSUVSGE
76	(4C) ADDRESS	4	BTVSGUP	->BLSUVSGU
80	(50) ADDRESS	4	BTVSPOP	->BLSUVSPO
84	(54) ADDRESS	4	BTVSPUP	->BLSUVSPU
88	(58) ADDRESS	20	BVT0023P	reserved
108	(6C) ADDRESS	4	BVTMONLP	->BLSUMONL
112	(70) ADDRESS	20	BVT0029P	reserved
132	(84) ADDRESS	4	BVTPUTCP	->BLSUPUTC
136	(88) ADDRESS	4	BVT0035P	reserved
140	(8C) ADDRESS	4	BVTZZ2RP	->BLSUZZ2R

BLSBVT - Storage Layout (continued)

Offsets	Type	Length	Name	Description
144	(90) ADDRESS	4	BVTPGMRP	->BLSUPGMR
148	(94) ADDRESS	4	BVTMONAR	->BLSUMONA
152	(98) ADDRESS	4	BVTBLDDP	->BLSUBLDD
156	(9C) ADDRESS	4	BVTBLDLP	->BLSUBLDL
160	(A0) ADDRESS	4	BVTPGMCP	->BLSUPGMC
164	(A4) ADDRESS	4	BVTPGMDP	->BLSUPGMD
168	(A8) ADDRESS	4	BVTPGMLP	->BLSUPGML
172	(AC) ADDRESS	8	BVT0044P	reserved
180	(B4) ADDRESS	4	BVTMONCP	->BLSUMONC
184	(B8) ADDRESS	4	BVT0047P	reserved
188	(BC) ADDRESS	4	BVTMONTP	->BLSUMONT
192	(C0) ADDRESS	4	BVTPARIP	->BLSUPARI
196	(C4) ADDRESS	4	BVTPARUP	->BLSUPARU
200	(C8) ADDRESS	8	BVT0051P	reserved
208	(D0) ADDRESS	4	BVTVSCR	->BLSUVSCR
212	(D4) ADDRESS	4	BVTVSMRP	->BLSUVSMR
216	(D8) ADDRESS	4	BVTMONXP	->BLSUMONX
220	(DC) ADDRESS	12	BVT0056P	reserved
232	(E8) ADDRESS	4	BVTPGMEP	->BLSUPGME
236	(EC) ADDRESS	4	BVTPGMSP	->BLSUPGMS
240	(F0) ADDRESS	4	BVTPGMUP	->BLSUPGMU
244	(F4) ADDRESS	4	BVTVSNBP	->BLSUVSNB
248	(F8) ADDRESS	4	BVTGNDMP	->BLSUGNDM
252	(FC) ADDRESS	4	BVTDDSNP	->BLSUDDSN
256	(100) ADDRESS	4	BVTMDSNP	->BLSUMDSN
260	(104) ADDRESS	4	BVTMOIIP	->BLSUMOII
264	(108) ADDRESS	4	BVTVASTP	->BLSUVAST
268	(10C) ADDRESS	4	BVTMONP	->BLSUMON
272	(110) ADDRESS	4	BVTINI3P	->BLSUINI3
276	(114) ADDRESS	4	BVTINI8P	->BLSUINI8
280	(118) ADDRESS	4	BVTSTKTP	->BLSUSTKT
284	(11C) ADDRESS	4	BVTSCM1P	->BLSUSCM1
288	(120) ADDRESS	4	BVTSTASP	->BLSUSTAS

BLSBVT - Storage Layout (continued)

<u>Offsets</u>	<u>Type</u>	<u>Length</u>	<u>Name</u>	<u>Description</u>
292 (124)	ADDRESS	4	BVTSTKDP	->BLSUSTKD
296 (128)	ADDRESS	4	BVTVARXP	->BLSUVARX
300 (12C)	ADDRESS	84	BVT0076P	reserved
384 (180)	CHARACTER	0	BVT99999	BLRVCT #MD83078

BLSDMCB

Common Name: Data Access Services Control Block

Macro ID: BLSDMCB

Created by: BLSCALOC

Size:

- 1844 bytes during allocation (includes the allocation work area)
- 648 bytes thereafter

Pointed to by:

- ZZ1PDRP (problem directory),
- ZZ1DSBP (data set directory),
- ZZ1DMCBP (DMCB chain anchor),
- DMCBNEXT (chain connector),
- parameter passed to data access services entry points

Function: Variables used by IPCS data access services to allocate and access a data set.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	648	DMCB	
0	(0) CHARACTER	4	DMCBID	DMCB IDENTIFIER DMCB-
4	(4) ADDRESS	4	DMCBNEXT	->NEXT DMCB ON CHAIN
8	(8) ADDRESS	4	DMCBTVP	->REQUESTOR TASK VARIABLE
12	(C) SIGNED	4	DMCBRTC	RETURN CODE
16	(10) SIGNED	4	DMCBRL	INPUT RECORD LENGTH
20	(14) BITSTRING	1	DMCBFTY	FILE TYPE
	1... ..		DMCBSHF	FILE SHARABLE FOR UPDATE
	.111 11..		DMCBFTRS	RESERVED
	1.		DMCBKSF	KEY SEQUENCED FILE
	1		DMCBVSF	VSAM FILE
21	(15) BITSTRING	1	DMCBOPN	OPEN FLAG
	1... ..		DMCBSIN	DMCB OPEN FOR INPUT
	.1..		DMCBSOUT	DMCB OPEN FOR OUTPUT
	..11 1111		DMCBRESA	RESERVED
22	(16) ADDRESS	1	DMCBRQST	CURRENT DATA ACC SERV REQ
23	(17) BITSTRING	1	DMCBRIOD	CURRENT REQUEST MODIFIER
	1... ..		DMCBOUT	OPEN FOR OUTPUT
	1... ..		DMCBTMP	TEMPORARY CLOSE
	.1..		DMCBRM6	DETERMINES CLOSE DISP WITH
				DMCBRM5.RM6=1->REREAD/REWIND.
				RM6=0->LEAVE/DISP
	..1.		DMCBRM5	RM5=1->REREAD/LEAVE. RM5=0->REWIND/DISP
	...1 1111		DMCBRMRS	RESERVED
24	(18) BITSTRING	1	DMCBOPTS	PROCESSING OPTION CODES
	1... ..		DMCBKYD	KEYED ACCESS
	.1..		DMCBGKY	GENERIC IF KEYED ACCESS
	.1..		DMCBRNO	RECORD NUMBER IF NOT KEYED
	..1.		DMCBAPX	INDICATES APPROXIMATE KEY

BLSDMCB - Storage Layout (continued)

Offsets	Type	Length	Name	Description
	...1		DMCBUPD	INDICATES GET FOR UPDATE
	 1...		DMCBBWD	INDICATES REVERSE SEQ
	1..		DMCBLRD	POINT TO LAST RECORD
	11		DMCBOPRS	RESERVED
25	(19) ADDRESS	1	DMCBLRQ	LAST DATA ACC SERV REQST
26	(1A) BITSTRING	1	DMCBLRM	LAST REQUEST MODIFIER
27	(1B) BITSTRING	1	DMCBLOPT	LAST PROCESSING OPTIONS
28	(1C) BITSTRING	1	DMCBMFLG	RESERVED
	1...		DMCBFMD	RESERVED
	.111 1111		DMCBRESB	RESERVED
29	(1D) UNSIGNED	1	DMCBREJ	REQUEST REJECTION REASON
30	(1E) UNSIGNED	1	DMCBACCM	MAX DATA ACCESS MODE
31	(1F) CHARACTER	1	DMCBRES1	RESERVED
32	(20) ADDRESS	4	DMCBBUFP	->DATA BUFFER
	1...		DMCBBFPC	CHANGE STATUS OF BUFPF
36	(24) SIGNED	4	DMCBBLEN	BUFFER LENGTH. NEGATIVE IF UNCHANGED
40	(28) SIGNED	4	DMCBORL	OUTPUT RECORD LENGTH. NEGATIVE IF UNCHANGED
44	(2C) ADDRESS	4	DMCBKEYP	->KEY
	1...		DMCBKPC	KEYP CHANGE STATUS
48	(30) SIGNED	4	DMCBKEYL	KEY LENGTH. NEGATIVE IF UNCHANGED
52	(34) ADDRESS	4	DMCBMSG	->MESSAGE AREA
56	(38) SIGNED	4	DMCBRBA	REL RECORD RBA
60	(3C) SIGNED	4	DMCBRRL	RECORD LENGTH FOR RELATIVE RECORD SUP- PORT. ZERO IF INELIGIBLE
64	(40) ADDRESS	4	DMCBRI	->BLSCRQST MODULE
68	(44) ADDRESS	4	DMCBMSG0	->BLSDMSG0 MODULE
72	(48) ADDRESS	4	DMCBMSG5	->MESSAGE TEXT
76	(4C) ADDRESS	4	DMCBZZ1P	->COMMON AREA
80	(50) ADDRESS	4	DMCBRESC	RESERVED
84	(54) UNSIGNED	1	DMCBSPID	DAS SUB-POOL ID
85	(55) CHARACTER	3	DMCBRES2	RESERVED
88	(58) ADDRESS	4	DMCBFRE	->NEXT FREE WORK AREA
92	(5C) ADDRESS	4	DMCBE0B	->END OF DMCB

Error Recovery Section

96	(60) CHARACTER	16	DMCBSTL	ESTAE PARAMETER LIST
112	(70) ADDRESS	4	DMCBRET	->ABEND RETRY
116	(74) UNSIGNED	4	DMCBCPC	ABEND COMPLETION CODE
120	(78) UNSIGNED	4	DMCBARC	ABEND RETURN CODE

BLSDMCB - Storage Layout (continued)

Offsets	Type	Length	Name	Description
124	(7C) UNSIGNED	4	DMCBACBE	ACBERROR FIELD AFTER OPEN
124	(7C) UNSIGNED	4	DMCBDCBE	REG 15 AT TIME OF OPEN OR CLOSE ABEND
128	(80) UNSIGNED	4	DMCBRPLF	RPL FDBK FIELD AFTER BAD VSAM REQUEST
132	(84) CHARACTER	128	DMCBSYNM	SYNAD MESSAGE

Allocation/Deallocation Section

260	(104) CHARACTER	8	DMCBDDNM	ALLOCATED DDNAME
268	(10C) CHARACTER	6	DMCBVOL	ALLOCATED VOLUME SERIAL
274	(112) CHARACTER	2	DMCBDSOR	ALLOCATED DATA SET ORG
276	(114) CHARACTER	8	DMCBUNIT	ALLOCATED UNIT NAME
284	(11C) CHARACTER	44	DMCBDSN	ALLOCATED DATA SET NAME
328	(148) CHARACTER	4	DMCBTYPE	DATA SET TYPE
332	(14C) CHARACTER	8	DMCBPID	PROBLEM ID
340	(154) CHARACTER	8	DMCBRES9	RESERVED
348	(15C) CHARACTER	8	DMCBMODL	ALLOCATION MODEL NAME
356	(164) CHARACTER	8	DMCBMEMB	ALLOCATED MEMBER NAME
364	(16C) CHARACTER	2	DMCBRES3	RESERVED
366	(16E) BITSTRING	1	DMCBSTAT	DATASET ALLOCATION STATUS
367	(16F) BITSTRING	1	DMCBDISP	DATASET NORMAL DISP
368	(170) BITSTRING	1	DMCBODIS	DATASET OVERRIDE DISP
369	(171) CHARACTER	3	DMCBSPTY	DATASET SPACE TYPE
372	(174) SIGNED	4	DMCBSPPR	DATASET PRI SPACE QUANT
376	(178) SIGNED	4	DMCBSPSE	DATASET SEC SPACE QUANT
380	(17C) BITSTRING	1	DMCBSPEC	DATASET SPACE SPECS
	1... ..		DMCBSPRL	DATASET SPACE RELEASE
	.1... ..		DMCBSPCT	DATASET SPACE CONTIGUOUS
	..1.		DMCBSPRN	DATASET SPACE ROUND
	...1 1111		DMCBSPRS	RESERVED
381	(17D) BITSTRING	1	DMCBAFLG	ALLOCATION FLAGS
	1... ..		DMCBCONF	CONNECT ALLOC REQST FLAG
	.iii iiii		DMCBRES4	RESERVED
382	(17E) BITSTRING	1	DMCBLBTY	DATASET LABEL TYPE
383	(17F) BITSTRING	1	DMCBLBPS	DATASET PASSWD PROT SPEC
384	(180) SIGNED	2	DMCBLBSQ	DATASET SEQUENCE NUMBER
386	(182) SIGNED	2	DMCBLBRT	DATASET RETENTION PERIOD
388	(184) BITSTRING	1	DMCBRES5	RESERVED
389	(185) BITSTRING	1	DMCBDCFM	DCB RECORD FORMAT
390	(186) SIGNED	2	DMCBDCLR	DCB LOGICAL RECORD LENGTH
392	(188) SIGNED	2	DMCBDCBL	DCB BLOCKSIZE
394	(18A) SIGNED	2	DMCBCISZ	VSAM CONTROL INTERVAL SIZE
396	(18C) SIGNED	2	DMCBRSZA	VSAM AVERAGE RECORD SIZE

BLSDMCB - Storage Layout (continued)

Offsets	Type	Length	Name	Description
398 (18E)	SIGNED	2	DMCBRSZM	VSAM MAXIMUM RECORD SIZE
400 (190)	ADDRESS	4	DMCBDRBP	->SVC 99 RB
404 (194)	UNSIGNED	4	DMCBSUBC	SVC 99 RETURN CODE
408 (198)	CHARACTER	2	DMCBIRSC	SVC 99 INFO/REASON CODE
410 (19A)	CHARACTER	2	DMCBRES6	RESERVED
412 (19C)	ADDRESS	4	DMCBFR	->BLSCFREE MODULE

Access Method Dependent Section

416 (1A0)	CHARACTER	152	DMCBAMS
-----------	-----------	-----	---------

Audit Trail Section

568 (238)	CHARACTER	64	DMCBAUDT	AUDIT TRAIL SECTION
568 (238)	ADDRESS	4	DMCBALC	RETURN ADDR OF ALLOCATOR
572 (23C)	ADDRESS	4	DMCBFRC	RETURN ADDR OF FREER
576 (240)	ADDRESS	4	DMCBOPC	RETURN ADDR OF OPENER
580 (244)	ADDRESS	4	DMCBCLC	RETURN ADDR OF CLOSER
584 (248)	ADDRESS	4	DMCBLSCP	->LAST ENTRY IN WRAP-AROUND LIST
588 (24C)	ADDRESS	44	DMCBCARY	WRAP-AROUND LIST
632 (278)	CHARACTER	8	DMCBMODN	ERROR MODULE NAME, FOR DAIRFAIL AND VSAMFAIL
640 (280)	CHARACTER	8	DMCBRES8	RESERVED

Common DMCB Trailer

648 (288)	CHARACTER	0	DMCBMRK	DATA ACC SERV WORK AREA
-----------	-----------	---	---------	-------------------------

VSAM Section

416 (1A0)	STRUCTURE	152	DMCBVSM	
416 (1A0)	CHARACTER	76	DMCBRPL	VSAM REQUEST PARAM LIST
492 (1EC)	CHARACTER	76	DMCBACB	VSAM ACC METH CNTL BLK

QSAM Section

BLSDMCB - Storage Layout (continued)

<u>Offsets</u>	<u>Type</u>	<u>Length</u>	<u>Name</u>	<u>Description</u>
416	(1A0) STRUCTURE	160	DMCBQSM	
416	(1A0) CHARACTER	96	DMCBDCB	QSAM DATA CONTROL BLOCK

QSAM DCB Exit List

512	(200) CHARACTER	8	DMCBXLST	DCB EXIT LIST
512	(200) ADDRESS	4	DMCBXLOP	OPEN EXIT
512	(200) BITSTRING	1	DMCBXLOC	OPEN EXIT CODE
513	(201) ADDRESS	3	DMCBXLOA	->OPEN EXIT
516	(204) ADDRESS	4	DMCBXLAB	ABEND EXIT
516	(204) BITSTRING	1	DMCBXLAC	ABEND EXIT CODE
517	(205) ADDRESS	3	DMCBXLAA	->ABEND EXIT
520	(208) CHARACTER	56	DMCBRES7	RESERVED

Allocation Work Area

648	(288) STRUCTURE	1196	DMAL	ALLOCATION WORK AREA
648	(288) CHARACTER	4	DMALID	DMAL IDENTIFIER
652	(28C) ADDRESS	4	DMALMODA	->ALLOCATION MODEL NAME
656	(290) ADDRESS	4	DMALPRMP	->ALLOCATION OVERRIDE PARAMETERS
660	(294) ADDRESS	4	DMALMODP	->ALLOCATION MODEL
664	(298) CHARACTER	20	DMALDYRB	DYNALLOC REQUEST BLOCK
684	(2AC) SIGNED	4	DMALTUPM	NUMBER OF VALID TU PTRS
688	(2B0) ADDRESS	4	DMALTUBS	->NEXT AVAILABLE SPACE IN DMALTUS
692	(2B4) ADDRESS	128	DMALTUPL	DYNALLOC TEXT UNIT POINTER LIST
820	(334) CHARACTER	1024	DMALTUS	DYNALLOC TEXT UNIT WORK AREA
1844	(734) CHARACTER	0	DMALEND	END OF ALLOCATION WORK AREA

Alphabetic Index to Field Names

<u>Name</u>	<u>Offset</u>	<u>Name</u>	<u>Offset</u>	<u>Name</u>	<u>Offset</u>
DMAL	288	DMALDYRB	298	DMALEND	734
DMALID	288	DMALMODA	28C	DMALMODP	294
DMALPRMP	290	DMALTUBS	2B0	DMALTUPL	2B4
DMALTUPM	2AC	DMALTUS	334	DMCB	0
DMCBACB	1EC	DMCBACBE	7C	DMCBACCM	1E
DMCBACFLG	17D	DMCBALC	238	DMCBAMIS	1A0
DMCBAPX	18	DMCBARC	78	DMCBAUDT	238
DMCBBFPC	20	DMCBLEN	24	DMCBBUFP	20
DMCBBMD	18	DMCBECARY	24C	DMCBCISZ	18A
DMCBCLC	244	DMCBCONF	17D	DMCBCPC	74
DMCBDCB	1A0	DMCBDCBE	7C	DMCBDCBL	188

BLSDMCB - Alphabetic Index to Field Names (continued)

Name	Offset		Name	Offset		Name		
DMCBDCFM	185		DMCBDCLR	186		DMCBDDNM	104	
DMCBDISP	16F		DMCBDRBP	190		DMCBDSN	11C	
DMCBDSOR	112		DMCBEOB	5C		DMCBFMOD	1C	80
DMCBFR	19C		DMCBFRC	23C		DMCBFRE	58	
DMCBFTRS	14	7C	DMCBFTY	14		DMCBGKY	18	40
DMCBID	0		DMCBIRL	10		DMCBIRSC	193	
DMCBKEYL	30		DMCBKEYP	2C		DMCBKPC	2C	80
DMCBKSF	14	02	DMCBKYD	18	80	DMCBLBPS	17F	
DMCBLBRT	182		DMCBLBSQ	180		DMCBLBTY	17E	
DMCBLOPT	1B		DMCBLRD	18	04	DMCBLRM	1A	
DMCBLRQ	19		DMCBLSCP	248		DMCBMEMB	164	
DMCBMFLG	1C		DMCBMODL	15C		DMCBMODN	278	
DMCBMSG	34		DMCBMSG5	48		DMCBMSG0	44	
DMCBNEXT	4		DMCBODIS	170		DMCBOPC	240	
DMCBOPN	15		DMCBOPRS	18	03	DMCBOPTS	18	
DMCBORL	28		DMCBOUT	17	80	DMCBPID	14C	
DMCBQSM	1A0		DMCBRBA	38		DMCBREJ	1D	
DMCBRESA	15	3F	DMCBRESB	1C	7F	DMCBRESC	50	
DMCBRES1	1F		DMCBRES2	55		DMCBRES3	16C	
DMCBRES4	17D	7F	DMCBRES5	184		DMCBRES6	19A	
DMCBRES7	208		DMCBRES8	280		DMCBRES9	154	
DMCBRET	70		DMCBRI	40		DMCBRMOD	17	
DMCBRMRS	17	1F	DMCBRM5	17	20	DMCBRM6	17	40
DMCBRHO	18	40	DMCBRPL	1A0		DMCBRPLF	80	
DMCBRQST	16		DMCBRRL	3C		DMCBRSZA	18C	
DMCBRSZM	18E		DMCBRTC	C		DMCBSHF	14	80
DMCBSIN	15	80	DMCBSOUT	15	40	DMCBSPCT	17C	40
DMCBSPEC	17C		DMCBSPID	54		DMCBSPPR	174	
DMCBSPRL	17C	80	DMCBSPRN	17C	20	DMCBSPRS	17C	1F
DMCBSPSE	178		DMCBSPTY	171		DMCBSTAT	16E	
DMCBSTL	60		DMCBSUBC	194		DMCBSYNM	84	
DMCBTMP	17	80	DMCBTVP	8		DMCBTYPE	148	
DMCBUNIT	114		DMCBUPD	18	10	DMCBVOL	10C	
DMCBVSF	14	01	DMCBVSM	1A0		DMCBWRK	288	
DMCBXLAA	205		DMCBXLAB	204		DMCBXLAC	204	
DMCBXLOA	201		DMCBXLOC	200		DMCBXLOP	200	
DMCBXLST	200		DMCBZZ1P	4C				

BLSDSD

Common Name: Data Set Directory Record Map

Macro ID: BLSDSD

Size:

- Base record = 164 bytes
- Problem association record = 68 bytes

Function: Identify problem related data sets known to IPCS (base record). Associate a data set with a specific problem known to IPCS (problem association record).

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	64	DSDREC	DATA SET DIRECTORY RECORD
0	(0) CHARACTER	61	DSDKEY	KEY OF DSD RECORD
0	(0) CHARACTER	53	DSDKGRP	GENERIC KEY OF DSD RECORD
0	(0) CHARACTER	1	DSDKTYP	DSD RECORD TYPE
1	(1) CHARACTER	52	DSDKDSNM	DATA SET NAME AND MEMBER
1	(1) CHARACTER	44	DSDKDSN	DATASET NAME
45	(2D) CHARACTER	8	DSDKMEMN	MEMBER NAME
53	(35) CHARACTER	8	DSDKBSID	DSD BASE RECORD ID
53	(35) CHARACTER	8	DSDKPID	DSD ASSOCIATED PROBLEM ID
53	(35) CHARACTER	3	DSDKPREF	DSD ASSOC PROB ID PREFIX
56	(38) CHARACTER	5	DSDKPNUM	DSD ASSOCIATED PROBLEM NUMBER
61	(3D) CHARACTER	3	DSDRES1	RESERVED, HEX ZEROS
64	(40) CHARACTER	0	DSDINFO	RECORD CONTENTS

Data Set Directory Base Record - Attributes

64	(40) STRUCTURE	80	DSDBASRC	DSD BASE RECORD
64	(40) CHARACTER	8	DSDDSNTY	DATA SET TYPE
72	(48) CHARACTER 1... ..	8	DSDDSATR DSDSLOIN	DATA SET ATTRIBUTES BIT TO DEFINE IF DS IS IPCS MANAGED
80	(50) CHARACTER	60	DSDDESC	DATA SET DESCRIPTION
140	(8C) CHARACTER	4	DSDBRES1	RESERVED

Data Set Directory Problem Association Record

64	(40) STRUCTURE	4	DSDPRASR	DSD PROBLEM ASSOCIATION RECORD
64	(40) CHARACTER	4	DSDPRSQN	SEQUENCE NUMBER OF THE PDR DATA SET RECORD THAT ASSOCIATES THIS DATA SET TO THE PROBLEM

BLSDVT

Common Name: Central Data Base Vector

Macro ID: BLSDSERV

Created by: Compilation of module BLSDVECT

Pointed to by: Field ZZ2DVTP (see “BLSUZZ2” on page 406)

Function: Provide the addresses of common subroutines and data shared by the IPCS subcomponents concerned with processing the problem and data set directories.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	256	DVT	BLRVCT #MD83078
0	(0) ADDRESS	8	DVT0001P	reserved
8	(8) ADDRESS	4	DVTDEQPP	->BLSDDQPP
12	(C) ADDRESS	4	DVTENQPP	->BLSDENQP
16	(10) ADDRESS	4	DVT0005P	reserved
20	(14) ADDRESS	4	DVTAUTHP	->BLSEAUTH
24	(18) ADDRESS	4	DVTLPCLP	->BLSELPCL
28	(1C) ADDRESS	4	DVTBP00P	->BLSFBP00
32	(20) ADDRESS	4	DVTCH00P	->BLSFCN00
36	(24) ADDRESS	4	DVTDF00P	->BLSFDF00
40	(28) ADDRESS	4	DVT0011P	reserved
44	(2C) ADDRESS	4	DVTDS00P	->BLSFDS00
48	(30) ADDRESS	4	DVTFL00P	->BLSFFL00
52	(34) ADDRESS	4	DVTFP00P	->BLSFFP00
56	(38) ADDRESS	4	DVTGD00P	->BLSFGD00
60	(3C) ADDRESS	4	DVTGG00P	->BLSFGG00
64	(40) ADDRESS	4	DVTGP00P	->BLSFGP00
68	(44) ADDRESS	4	DVTND00P	->BLSFND00
72	(48) ADDRESS	4	DVTND01P	->BLSFND01
76	(4C) ADDRESS	4	DVTOD00P	->BLSFOD00
80	(50) ADDRESS	4	DVTPS00P	->BLSFPS00
84	(54) ADDRESS	4	DVTSD00P	->BLSFSD00
88	(58) ADDRESS	4	DVTSL00P	->BLSFSL00
92	(5C) ADDRESS	4	DVTTL00P	->BLSFTL00

BLS DVT - Storage Layout (continued)

Offsets	Type	Length	Name	Description
96	(60) ADDRESS	4	DVTUD00P	->BLSFUD00
100	(64) ADDRESS	4	DVTUP00P	->BLSFUP00
104	(68) ADDRESS	4	DVTL P00P	->BLSFLP00
108	(6C) ADDRESS	4	DVTLALLP	->BLSFLALL
112	(70) ADDRESS	4	DVTL PFMP	->BLSELPFM
116	(74) ADDRESS	4	DVTCLOSP	->BLSFCLOS
120	(78) ADDRESS	4	DVTO PENP	->BLSFOPEN
124	(7C) ADDRESS	4	DVTPERAP	->BLSFPERA
128	(80) ADDRESS	4	DVTDALLP	->BLSFDALL
132	(84) ADDRESS	124	DVT0034P	reserved
256	(100) CHARACTER	0	DVT99999	BLRVCT #MD83078

BLSFP

Common Name: Facility Parameters Block

Macro ID: BLSFP

Created by: BLSDIFP0

Size: 152 bytes

Pointed to by: Field ZZ1FPP (see “BLSUZZ1” on page 404)

Function: Describe IPCS facility parameters obtained from SYS1.PARMLIB(IPCSPRnn).

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	152	FPBLOK	FACILITY PARAMETERS BLOCK
0	(0) CHARACTER	44	FPPDDS	PROBLEM DIRECTORY DATA SET NAME
44	(2C) CHARACTER	4	FPRES1	RESERVED
48	(30) CHARACTER	44	FPDSDS	DATA SET DIRECTORY DATA SET NAME
92	(5C) CHARACTER	4	FPRES2	RESERVED
96	(60) CHARACTER	3	FPPIDP	DEFAULT PROBLEM ID PREFIX
99	(63) CHARACTER	1	FPRES3	RESERVED
100	(64) CHARACTER	3	FPDBPIDP	DEFAULT DATA BASE PROB ID PRE- FIX--EBCDIC 000
103	(67) CHARACTER	1	FPRES4	RESERVED
104	(68) CHARACTER	8	FPSYSID	SYSTEM IDENTIFICATION
112	(70) CHARACTER	8	FPGRPID	GROUP IDENTIFICATION
120	(78) CHARACTER	8	FPADMNM	ID OF PERSON HAVING IPCS ADMIN AUTHORI- TY
120	(78) CHARACTER	7	FPADMID	TSOID OF PERSON HAVING IPCS ADMINSTRA- TIVE AUTHORITY
127	(7F) ADDRESS	1	FPADMLN	LENGTH OF ADMIN TSO ID
128	(80) CHARACTER	8	FPDELMN	ID OF PERSON HAVING IPCS DATA SET DELETE AUTH
128	(80) CHARACTER	7	FPDELID	TSOID OF PERSON HAVING IPCS DATA SET DELETE AUTHORITY
135	(87) ADDRESS	1	FPDELLN	LENGTH OF DELETE TSO ID
136	(88) CHARACTER	16	FPRES6	RESERVED

BLSLBLS

Common Name: BLSLDISP Dialog Communication Block

Macro ID: BLSLBLS

Created by: ISPF dialog program BLSLDISP

Pointed to by: Parameter passed by BLSLDISP

Function: Communicate among ISPF dump display dialog modules.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	6144	BLS	BLSLDISP dialog block
0	(0) CHARACTER	8	BLSCTL	Block control data
0	(0) CHARACTER	7	BLSID	Identifier=>BLSLBLS
7	(7) UNSIGNED	1	BLSLVL	Change level=>'02'X

Miscellany

8	(8) CHARACTER	66	BLSAD	Default dump source for BLSLDISP dialog
74	(4A) CHARACTER	6	BLRSV01	reserved
80	(50) CHARACTER	16	BLSAQAS	Default address space
96	(60) BITSTRING	8	BLSF	Control flags
	1... ..		BLSFCAN	CANCEL primary command
	.1... ..		BLSFPOSM	Floating stack position
	..11 1111			
	1111 1111			
	1111 1111			
	1111 1111			
	1111 1111			
	1111 1111			
	1111 1111			
	1111 1111			
	1111 1111			reserved
104	(68) CHARACTER	5	BLSTACK0	Stack display origin
109	(6D) CHARACTER	219	BLRSV02	reserved

Variables Defined to ISPF

328	(148) CHARACTER	250	BLSCMDIN	Primary command text
578	(242) CHARACTER	250	BLSPVAL	Pointer value
828	(33C) CHARACTER	12	BLRSV03	reserved
840	(348) CHARACTER	130	BLSDDMP	Dump data set names
840	(348) CHARACTER	65	BLSDSRC	Default dump name
905	(389) CHARACTER	65	BLSOSRC	Override dump name
970	(3CA) CHARACTER	44	BLSDC22	Address space names
970	(3CA) CHARACTER	22	BLSDAS	Default address space

BLSLBLS - Storage Layout (continued)

Offsets	Type	Length	Name	Description
992 (3E0)	CHARACTER	22	BLSOAS	Override address space
1014 (3F6)	CHARACTER	2	BLSRSV04	reserved
1016 (3F8)	CHARACTER	56	BLSCRM5	Member/variable names
1016 (3F8)	CHARACTER	8	BLSCRSRF	Cursor field name
1024 (400)	CHARACTER	8	BLSMSG	Message identifier
1024 (400)	CHARACTER	5	BLSMSGA	Common prefix
1029 (405)	CHARACTER	3	BLSMSGB	Message identifier suffix
1029 (405)	CHARACTER	2	BLSMSGBA	Library member suffix
1031 (407)	CHARACTER	1	BLSMSGB3	Message number in member
1032 (408)	CHARACTER	8	BLSPSWD	Password
1040 (410)	CHARACTER	8	BLSFSER	Failing service name
1048 (418)	CHARACTER	8	BLSVERB	ZVERB buffer
1056 (420)	CHARACTER	16		reserved
1072 (430)	CHARACTER	40	BLSFIXED	Numeric fields
1072 (430)	UNSIGNED	4	BLSCRPOS	Cursor position
1076 (434)	UNSIGNED	4	BLSWIDTH	Area line width
1080 (438)	UNSIGNED	4	BLSDEPTH	Area depth in lines
1084 (43C)	UNSIGNED	4	BLSLVLIN	Last visible line
1088 (440)	UNSIGNED	4	BLSCROLL	ZSCROLLN buffer
1092 (444)	UNSIGNED	20		reserved
1112 (458)	CHARACTER	4	BLSCRTYP	ZSCROLLA buffer
1116 (45C)	BITSTRING	4	BLSSRC	Service return code

Dump Directory Data Buffers

1120 (460)	CHARACTER	694	BLSESBUF	Symbol table record buffer
1814 (716)	CHARACTER	234	BLSRSV05	reserved
2048 (800)	CHARACTER	3072	BLSVSBUF	Sequential dump directory I/O buffer
5120 (1400)	CHARACTER	238	BLSRSV06	reserved
5358 (14EE)	CHARACTER	250	BLSFINDV	Text--Find search value
5608 (15E8)	CHARACTER	536	BLSFD	Find data

Detailed Structure of Field BLSESBUF

1120 (460)	STRUCTURE	694	BLSES	BLSRESSY BUFFER MF (LIST,BLSES,DEFINED(BLSESBUF))
1120 (460)	CHARACTER	2	BLSESRID	Record type==>ES
1122 (462)	BITSTRING	6		BLSURISY #MD82026

BLSLBS - Storage Layout (continued)

Offsets	Type	Length	Name	Description
1128 (468)	CHARACTER	8		
1136 (470)	BITSTRING	8		
1144 (478)	UNSIGNED	4	BLSESRDX	Data set index
1148 (47C)	CHARACTER	31	BLSESSYM	Equated symbol
1179 (49B)	CHARACTER	0	BLSESELK	End of logical key
1179 (49B)	CHARACTER	1	BLSESRV1	reserved
1180 (49C)	CHARACTER	16	BLSESAS	BLSRDATS #MD83116
1180 (49C)	CHARACTER	0	BLSESAS0	BLSRDATS #MD83116
1180 (49C)	CHARACTER	2	BLSESASt	Address space type code
1182 (49E)	BITSTRING	2		
1184 (4A0)	UNSIGNED	4	BLSESAS1	Integer 1
1188 (4A4)	UNSIGNED	4	BLSESAS2	Integer 2
1192 (4A8)	BITSTRING	4		
1196 (4AC)	CHARACTER	0	BLSESAS9	BLSRDATS #MD83116
1196 (4AC)	ADDRESS	4	BLSESLAD	Logical address
1200 (4B0)	CHARACTER	60	BLSESD	BLSRDATC MF(SUBLIST,BLSESD,,2)
1200 (4B0)	CHARACTER	0	BLSESD00	BLSRDATC #MD83116
1200 (4B0)	SIGNED	4	BLSESDOF	Offset in bytes
1204 (4B4)	UNSIGNED	4	BLSESDLE	Length in bytes
1208 (4B8)	UNSIGNED	1	BLSESD0B	
1209 (4B9)	UNSIGNED	1	BLSESDLB	
1210 (4BA)	CHARACTER	34	BLSESDT	BLSRDATT MF(SUBLIST,BLSESDT,,3)
1210 (4BA)	CHARACTER	0	BLSESDT0	BLSRDATT #MD83116
1210 (4BA)	CHARACTER	1	BLSESDTY	Data type code
1211 (4BB)	BITSTRING	1		
1212 (4BC)	CHARACTER	31	BLSESDTD	Data name
1243 (4DB)	CHARACTER	1	BLSESDTE	reserved
1244 (4DC)	CHARACTER	0	BLSESDT9	BLSRDATT #MD83116
1244 (4DC)	UNSIGNED	4	BLSESDIM	Array entry count
1248 (4E0)	SIGNED	4	BLSESDIL	Subscript of initial array entry
1252 (4E4)	BITSTRING	4	BLSESDF BLSESDFA	Flags Array
	1111 1111 1111 1111 1111 1111 1111 1111			

BLSLBLS - Storage Layout (continued)

Offsets	Type	Length	Name	Description
1256	(4E8) BITSTRING	4		
1260	(4EC) CHARACTER	0	BLSESD99	BLSRDATC #MD83116
1260	(4EC) ADDRESS	4	BLSESMAD	->1st byte missing
1264	(4F0) BITSTRING	16		

System/370 storage key summary--the storage key(s) for the main storage described by this record are summarized by:

1. Taking the storage key for the first 2K block
2. Setting the reference and change bits in the key if they are on in any 2K block between the origin of the storage described by the symbol and the first storage not available in the dump.

1280	(500) BITSTRING	1	BLSESKEY	System/370 storage key. All bits on when indeterminate
------	-----------------	---	----------	--

Flags

1281	(501) BITSTRING	3	BLSESF	Flags
1281	(501) BITSTRING	1	BLSESF5	Storage flags
	1... ..		BLSESF5C	Storage information complete
	.1.. ..		BLSESF52	Multiple storage keys
	..1.		BLSESF5M	Not all storage in dump
	...1		BLSESF5A	Absolute address, field BLSESABS, is valid
	 1...		BLSESF5P	Prefixed storage
	1..		BLSESF5R	Reclaimed storage
	1.		BLSESF5X	Multiple records
	1		BLSESF5S	SUMDUMP data
1282	(502) BITSTRING	1	BLSESF5C	Control flags
	1... ..		BLSESF5CD	Drop not permitted
	.111 1111			
1283	(503) BITSTRING	1	BLSESF5C0	Common storage
	1... ..			
	.111 1111			
1284	(504) UNSIGNED	4	BLSESABS	Absolute address for this address
1288	(508) BITSTRING	12		

Remark--LENGTH(0:512) characters

1300	(514) CHARACTER	514	BLSESRL	BLRPRM #MD82025
1300	(514) UNSIGNED	2	BLSESRL	Length of BLSESRT
1302	(516) CHARACTER	512	BLSESRT	BLRTEXT #MD82080
1302	(516) CHARACTER	512	BLSESRTA	Section 1
1814	(716) CHARACTER	0	BLSES999	BLRESSY #MD83116

Detailed Structure of Field BLSAQAS

80	(50) STRUCTURE	16	BLSAS	BLSRDATS #MD83116
80	(50) CHARACTER	0	BLSAS0	BLSRDATS #MD83116

An IPCS address space is specified by three values:

1. A two-character code identifying the type of address space:
 - Letter code A indicates that the file contains an MVS high-speed dump and that absolute main storage of the dumped system is being referenced (ABSOLUTE)
 - Code BL indicates that a physical block in a file is being referenced using a relative block number (BLOCK)
 - Code BS indicates that a relative byte address group in the file is being referenced (RBA)
 - Code BT indicates that a physical block in a file is being referenced using a relative track address (TTR)
 - Letter code C indicates that the file contains an MVS high-speed dump and that the CPU status data for one dumped CPU is being referenced (STATUS)
 - Code CR indicates that the file contains an MVS high-speed dump and that real main storage seen by one CPU is being referenced (REAL)
 - Code CV indicates that the file contains an MVS high-speed dump and that virtual main storage seen by one MVS address space dispatched on a designated CPU is being referenced (ASID)
 - Letter code H indicates that the file contains an MVS high-speed dump and that the header data for the dump is being referenced (HEADER)
2. A binary integer whose interpretation depends on the preceding code:
 - For letter codes A and H this integer has no meaning and should be set to zero.
 - For code BL this integer should be the relative block number
 - For code BS this integer should be the relative byte address group number
 - For code BT this integer should be the relative track address
 - For codes beginning with the letter C this integer contains the System/370 CPU address (STAP instruction) for the referenced CPU.
3. A binary integer whose interpretation depends on the preceding code:
 - For code CV this integer contains the address space identification (ASID) for the referenced address space.
 - For other codes this integer has no meaning and should be set to zero.

80	(50) CHARACTER	2	BLSAST	Address space type code
82	(52) BITSTRING	2		
84	(54) UNSIGNED	4	BLSAS1	Integer 1

Offsets	Type	Length	Name	Description
88	(58) UNSIGNED	4	BLSAS2	Integer 2
92	(5C) BITSTRING	4		
96	(60) CHARACTER	0	BLSAS9	BLSRDATS #MD83116

Alphabetic Index to Field Names

Name	Offset		Name	Offset		Name	
BLS	0		BLSAD	8		BLSAQAS	50
BLSAS	50		BLSAST	50		BLSAS0	50
BLSAS1	54		BLSAS2	58		BLSAS9	60
BLSCMDIN	148		BLSCRMS	3F8		BLSCROLL	440
BLSCRPOS	430		BLSCRSRF	3F8		BLSCRTYP	458
BLSCCTL	0		BLSDAS	3CA		BLSDC22	3CA
BLSDTMP	348		BLSDPTH	438		BLSDSRC	348
BLSES	460		BLSESABS	504		BLSESAS	49C
BLSESAST	49C		BLSESAS0	49C		BLSESAS1	4A0
BLSESAS2	4A4		BLSESAS9	4AC		BLSESBUF	460
BLSESD	480		BLSESDF	4E4		BLSESDFA	4E4
BLSESDIL	4E0		BLSESDIM	4DC		BLSESDLB	4B9
BLSESDLE	4B4		BLSESDOB	4B8		BLSESDOF	4B0
BLSESDT	4BA		BLSESDTD	4BC		BLSESDTE	4DB
BLSESDTY	4BA		BLSESDT0	4BA		BLSESDT9	4DC
BLSESD00	4B0		BLSESD99	4EC		BLSESELK	49B
BLSESF	501		BLSESFC	502		BLSESFCD	502
BLSESFC0	503	80	BLSESFS	501		BLSESFSA	501
BLSESFSC	501	80	BLSESFSM	501	20	BLSESFSP	501
BLSESFSR	501	04	BLSESFSS	501	01	BLSESFSX	501
BLSESFS2	501	40	BLSESFKEY	500		BLSESLAD	4AC
BLSESMAD	4EC		BLSESR	514		BLSESRDX	478
BLSESRID	460		BLSESRJ	514		BLSESRV	516
BLSESRV0	516		BLSESRV1	49B		BLSESSYM	47C
BLSESRV99	716		BLSF	60		BLSFICAN	60
BLSF	15E8		BLSFINDV	14EE		BLSFIXED	430
BLSFPOS	60	40	BLSFSE	410		BLSID	0
BLSLVL	7		BLSLVLIN	43C		BLMSG	400
BLMSG	400		BLMSGB	405		BLMSGBA	405
BLMSGBB	407		BLSOAS	3E0		BLSSRC	389
BLSPSMD	408		BLSPVAL	242		BLSSRV01	4A
BLSSRV02	6D		BLSSRV03	33C		BLSSRV04	3F6
BLSSRV05	716		BLSSRV06	1400		BLSSRC	45C
BLSTACK0	68		BLSVRB	418		BLSVSBUF	800
BLSWIDTH	434						

BLSLNTRC

Common Name: Intercept Dialog Block

Macro ID: BLSLNTRC

Created by: BLSUTRMV (NTRC) and BLSLSTRG (NTRCPROG)

Pointed to by: Field ZZ2NTRCP (see "BLSUZZ2" on page 406)

Function: Communicate among modules that intercept line mode terminal output and present that output using ISPF full screen panels.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	1312	NTRC	BLSLMON dialog communication block
0	(0) UNSIGNED	4	NTRCFLNE	First line of line command block
4	(4) ADDRESS	4	NTRCNXTL	Sequentially, the next valid line for line command because of previous command amount
8	(8) UNSIGNED	4	NTRCPSZE	Holds size of last display area storage for freemain
12	(C) UNSIGNED	4	NTRCOUNT	Used to count number of lines in line command block
16	(10) UNSIGNED	4	NTRCSAMT	Number of lines collected for next screenful
20	(14) SIGNED	4	NTRCCPOS	Cursor position after display
24	(18) SIGNED	4	NTRCBCOL	Keeps track of requested beginning column of display
28	(1C) CHARACTER	8	NTRCDTYP	Panel area type
36	(24) CHARACTER	16	NTRCQUERY	Variables returned from PQUERY
36	(24) SIGNED	4	NTRCDWTH	Number of columns in dynamic area
40	(28) SIGNED	4	NTRCDPTH	Number of rows in area
44	(2C) SIGNED	4	NTRCRNUM	Row of top left position of area
48	(30) SIGNED	4	NTRCCNUM	Column for top left position of area
52	(34) ADDRESS	4	NTRCSBGN	Beginning of cell pool
56	(38) ADDRESS	4	NTRCSTRT	Addr in data storage where cells start after forward pointer
60	(3C) SIGNED	4	NTRCSEND	The ending address of current storage block
64	(40) ADDRESS	4	NTRCSTOR	Gets address of each storage cell
68	(44) ADDRESS	4	NTRCDSLP	Points to current line of display storage

BLSLNTRC - Storage Layout (continued)

Offsets	Type	Length	Name	Description
72	(48) ADDRESS	4	NTRCDSTG	Points to the beginning of Get mained storage for display
76	(4C) ADDRESS	4	NTRCPTOP	Previous top row pointer for table display
80	(50) ADDRESS	4	NTRCBTBM	Points to the Bottom of Data line on display
84	(54) ADDRESS	4	NTRCPREV	Used for setting the current backptr to previous line
88	(58) ADDRESS	4	NTRCQPTR	Initially for work ptr, then holds beginning of line command block
92	(5C) ADDRESS	4	NTRCLPTR	Used to point to last line of line command block
96	(60) ADDRESS	4	NTRCSTRL	Points to storage pointer in previous getmained storage
100	(64) ADDRESS	4	NTRCAPTR	Points to first cell of available storage
104	(68) ADDRESS	4	NTRCELLQ	Current cell address for FIND
108	(6C) ADDRESS	4	NTRCVARQ	Cell's offset for FIND search
112	(70) ADDRESS	4	NTRCSAVT	Save the top display line in case FIND data not found
116	(74) CHARACTER	32		Reserved
148	(94) CHARACTER 1... ..	8	NTRCFLGS NTRCXCLB	When ZZZ1, beginning of exclude block found
	.1... ..		NTRCDELB	When ZZZ1, beginning of delete block found
	..1.		NTRCCMPL	
	...1		NTRCFULL	Storage full when on
	 1...		NTRCERR	A syntax error found
	1..		NTRCDAMT	When on, previous command had a delete amount
	1.		NTRCEAMT	Previous command had an exclude amount
	1		NTRCLCMD	Previous command was followed by a line command
	1... ..		NTRCFINL	A syntax error found
	.1... ..		NTRCPAS1	Signals first pass for setups
	..1.		NTRCMORE	New display screen starting
	...1		NTRCFBTM	Hit bottom of data in FIND
	 1...		NTRCFXCL	Search excluded lines in FIND processing
	1..		NTRCFNXL	Search non-excluded lines in FIND processing
	1.		NTRCFNDC	If zero, no find command entered yet.
	1		NTRCTOP	RFIND is invalid
				Used to signal FIND search should start at top
150	(96) BITSTRING	6		Reserved
156	(9C) CHARACTER	8	NTRCMSG	Error message to be displayed with next table display
164	(A4) CHARACTER	250	NTRCCMD	Input command
414	(19E) CHARACTER	250	NTRCSTNG	Save original FIND command for use in panel message

BLSLNTRC - Storage Layout (continued)

Offsets	Type	Length	Name	Description
664 (298)	CHARACTER	8		Reserved
672 (2A0)	CHARACTER	536	NTRCDFD	FIND information
1208 (4B8)	CHARACTER	8	NTRCCFLD	Cursor field
1216 (4C0)	CHARACTER	96	NTRCFMSG	FIND string for message

Describes detailed structure of field NTRCDFD

672 (2A0)	STRUCTURE	536	FD	BLSRFD #MD83088
-----------	-----------	-----	----	-----------------

Value Sought

672 (2A0)	CHARACTER	260	FDSE	BLSRVALY #MD83084
672 (2A0)	CHARACTER	0	FDSE0	BLSRVALY #MD83084
672 (2A0)	CHARACTER	1	FDSET	Data type code
673 (2A1)	BITSTRING	1		
674 (2A2)	UNSIGNED	2	FDSEL	Length of value (bytes)
676 (2A4)	CHARACTER	256	FDSEC	EBCDIC value
676 (2A4)	BITSTRING	256	FDSEX	Hexadecimal value
676 (2A4)	SIGNED	4	FDSEF	Fullword numeric value
676 (2A4)	SIGNED	2	FDSEH	Halfword numeric value
678 (2A6)	SIGNED	2		
680 (2A8)	BITSTRING	252		
932 (3A4)	CHARACTER	0	FDSE9	BLSRVALY #MD83084

Mask Value

932 (3A4)	CHARACTER	260	FDMA	BLSRVALY #MD83084
932 (3A4)	CHARACTER	0	FDMA0	BLSRVALY #MD83084
932 (3A4)	CHARACTER	1	FDMAT	Data type code
933 (3A5)	BITSTRING	1		
934 (3A6)	UNSIGNED	2	FDMAL	Length of value (bytes)
936 (3A8)	CHARACTER	256	FDMAC	EBCDIC value
936 (3A8)	BITSTRING	256	FDMAX	Hexadecimal value
936 (3A8)	SIGNED	4	FDMAF	Fullword numeric value
936 (3A8)	SIGNED	2	FDMAH	Halfword numeric value
938 (3AA)	SIGNED	2		
940 (3AC)	BITSTRING	252		

BLSLNTRC - Storage Layout (continued)

Offsets	Type	Length	Name	Description
1192 (4A8)	CHARACTER	0	FDMA9	BLSRVALY #MD83084
Boundary Alignment Required				
1192 (4A8)	UNSIGNED	4	FDBDY	Boundary
1196 (4AC)	UNSIGNED	4	FDOFF	Offset from boundary
1200 (4B0)	UNSIGNED	4		
Additional Search Options				
1204 (4B4)	BITSTRING	3	FDF FDFBREAK	Search options Break option
	1... .. .111 1111 1111 1111 1111 1111			
1207 (4B7)	UNSIGNED	1	FDRELATE	Relationship sought
1208 (4B8)	CHARACTER	0	FD99999	BLSRFD #MD83088
0 (0)	STRUCTURE	284	NTRCPRCG	Storage cell mapping
0 (0)	CHARACTER	276	NTRCDATA	Special chars and data
0 (0)	CHARACTER	4	NTRCTYPE NTRCCHNG NTRCMSGL NTRCDISP NTRCFRST NTRCLAST NTRCFNDL	Marks special types of data Contains line command Exclude message line ZZZ1-display, ZZZ0-exclude Top of Data line Bottom of Data line Contains FIND data to display
	1... .. .1..1.1 1...1..11 1111 1111 1111 1111 1111 1111			
				Reserved
4 (4)	UNSIGNED	4	NTRCXAMT	Only for exclude message lines. Number excluded in this block.
8 (8)	UNSIGNED	4	NTRCLAMT	Saves a line amount entered after syntax check
12 (C)	CHARACTER	6	NTRCINPT	Stores any input line commands. Blank when none entered
18 (12)	CHARACTER	1	NTRCCHAR	Marks special types of data
19 (13)	CHARACTER	256	NTRCPVAR	Present line from the table
275 (113)	UNSIGNED	1	NTRCLGTH	Length of displayable data in the line
276 (114)	CHARACTER	8	NTRCPTRS	
276 (114)	ADDRESS	4	NTRCBPTR	Points to previous cell in cell pool
280 (118)	ADDRESS	4	NTRCFPTR	Points to next cell in cell pool

Alphabetic Index to Field Names

Name	Offset		Name	Offset		Name		
FD	2A0		FDBDY	4A8		FDF	4B4	
FDFBREAK	4B4	80	FDMA	3A4		FDIAC	3A8	
FDMAF	3A8		FDMAH	3A8		FDIAL	3A6	
FDMAT	3A4		FDMAX	3A8		FDMA0	3A4	
FDMA9	4A8		FDOFF	4AC		FDRELATE	4B7	
FDSE	2A0		FDSEC	2A4		FDSEF	2A4	
FDSEH	2A4		FDSEL	2A2		FDSET	2A0	
FDSEX	2A4		FDSE0	2A0		FDSE9	3A4	
FD99999	4B8		NTRC	0		NTRCAPTR	64	
NTRCBCOL	18		NTRCBPTR	114		NTRCBTBM	50	
NTRCCFLD	4B8		NTRCCHAR	12		NTRCCHNG	0	80
NTRCCMD	A4		NTRCCMPL	94	20	NTRCCNUM	30	
NTRCCPOS	14		NTRCDAMT	94	04	NTRCDATA	0	
NTRCDELB	94	40	NTRCDFD	2A0		NTRCDISP	0	20
NTRCDPTH	28		NTRCDSLP	44		NTRCDSTG	48	
NTRCDTYP	1C		NTRCDINTH	24		NTRCEAMT	94	02
NTRCELLA	68		NTRCERR	94	08	NTRCFBIM	95	10
NTRCFINL	95	80	NTRCFLGS	94		NTRCFLNE	0	
NTRCFMSG	4C0		NTRCFNDC	95	02	NTRCFNDL	0	04
NTRCFNXL	95	04	NTRCFPTR	118		NTRCFRST	0	10
NTRCFULL	94	10	NTRCFXCL	95	08	NTRCINPT	C	
NTRCLAMT	8		NTRCLAST	0	08	NTRCLCMD	94	01
NTRCLGTH	113		NTRCLPTR	5C		NTRCMORE	95	20
NTRCMSG	9C		NTRCMSGL	0	40	NTRCNXTL	4	
NTRCOUNT	C		NTRCPAS1	95	40	NTRCPREV	54	
NTRCPROG	0		NTRCPSZE	8		NTRCPTOP	4C	
NTRCPTRS	114		NTRCPVAR	13		NTRCQPTR	58	
NTRCQURY	24		NTRCRNUM	2C		NTRCSAMT	10	
NTRCSAVT	70		NTRCSEGN	34		NTRCSEND	3C	
NTRCSTNG	19E		NTRCSTOR	40		NTRCSTRL	60	
NTRCSTRT	38		NTRCTOP	95	01	NTRCTYPE	0	
NTRCVAR0	6C		NTRCXAMT	4		NTRCXCLB	94	80

BLSPDR

Common Name: Problem Directory Record Map

Macro ID: BLSPDR

Size:

- Problem status record type = 416 bytes
- Problem description record = variable 88-952 bytes
- Data set association record = 80 bytes
- PDR seed record = 416 bytes

Function: Contains:

- Status information for a problem reported to IPCS (problem status record)
- A textual description of a problem reported to IPCS (problem description record)
- A list of data sets containing problem-related data with a problem reported to IPCS (data set association record).
- The next IPCS problem number to be assigned (PDR seed record)

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	16	PDREC	PROBLEM DIRECTORY RECORD
0	(0) CHARACTER	14	PDRKEY	PROBLEM DIR KEY THIS RECORD
0	(0) CHARACTER	10	PDRGRP	PROBLEM DIR GENERIC KEY
0	(0) CHARACTER	2	PDRKRCTY	PDR RECORD TYPE
2	(2) CHARACTER	8	PDRKPID	PDR PROBLEM ID
2	(2) CHARACTER	3	PDRKPREF	PREFIX
5	(5) CHARACTER	5	PDRKPNUM	PROBLEM NUMBER OF RECORD
10	(A) CHARACTER	4	PDRRECSQ	SEQ NUMBER OF RECORD
14	(E) CHARACTER	2	PDRRESV	RESERVED
16	(10) CHARACTER	0	PDRINFO	RECORD CONTENT THIS PDR

Problem Status Record Section

16	(10) STRUCTURE	400	PDRSTREC	STATUS RECORD
16	(10) CHARACTER	8	PDRDATE	DATE OF PROB OCCURANCE
24	(18) CHARACTER	8	PDRTIME	TIME OF PROBLEM OCCURANCE
32	(20) CHARACTER	10	PDRCOMP	SUSPECTED FAILING COMPONENT ID
42	(2A) CHARACTER	1	PDRSEV	SEVERITY OF PROBLEM
43	(2B) CHARACTER	5	PDRRES1	RESERVED HEX ZEROS
48	(30) CHARACTER	8	PDROWNNM	NAME OF PERSON RESPONSIBLE FOR PROB
48	(30) CHARACTER	7	PDROWNID	TSOID OF PERSON RESPONSIBLE FOR PROB
55	(37) ADDRESS	1	PDROWNLN	PROBLEM OWNER TSOID LENGTH

BLSPDR - Storage Layout (continued)

Offsets	Type	Length	Name	Description
56	(38) CHARACTER	8	PDRGROUP	GROUP RESPONSIBLE FOR PROBLEM
64	(40) CHARACTER	8	PDRSYS	SYSTEM ON WHICH PROBLEM OCCURED
72	(48) CHARACTER	8	PDRUSER	USER/INSTALLATION DEFINED FIELD
80	(50) CHARACTER	8	PDRPSTAT	PROBLEM STATUS
88	(58) CHARACTER	8	PDRPDATE	DATE OF PROBLEM STATUS CHANGE
96	(60) CHARACTER	8	PDRPTIME	TIME OF PROBLEM STATUS CHANGE
104	(68) CHARACTER	8	PDRRDATE	DATE PROBLEM REPORTED
112	(70) CHARACTER	8	PDRRTIME	TIME PROBLEM REPORTED
120	(78) CHARACTER	8	PDRISTAT	IBM STATUS
128	(80) CHARACTER	8	PDRIDATE	DATE OF IBM STATUS CHANGE
136	(88) CHARACTER	8	PDRITIME	TIME OF IBM STATUS CHANGE
144	(90) CHARACTER	7	PDRAPRID	APAR IDENTIFICATION
151	(97) CHARACTER	5	PDRRES3	RESERVED HEX ZEROS
156	(9C) CHARACTER	4	PDRTSTAT	PTF STATUS
160	(A0) CHARACTER	8	PDRTDATE	DATE OF PTF STATUS CHANGE
168	(A8) CHARACTER	8	PDRTTIME	TIME OF PTF STATUS CHANGE
176	(B0) CHARACTER	7	PDRPTFID	PTF IDENTIFICATION
183	(B7) CHARACTER	5	PDRRES4	RESERVED HEX ZEROS
188	(BC) CHARACTER	4	PDRFSTAT	FIX STATUS
192	(C0) CHARACTER	8	PDRFDATE	DATE OF FIX STATUS CHANGE
200	(C8) CHARACTER	8	PDRFTIME	TIME OF FIX STATUS CHANGE
208	(D0) CHARACTER	60	PDRFIXID	FIX IDENTIFICATION
268	(10C) CHARACTER	4	PDRRES5	RESERVED HEX ZEROS
272	(110) CHARACTER	128	PDRABS	ABSTRACT OF PROBLEM
400	(190) SIGNED	4	PDRDESCL	COUNT OF TOTAL DESCR LINES
404	(194) SIGNED	4	PDRTRKL	COUNT OF TOTAL OTB DATA LINES
408	(198) CHARACTER	4	PDRRES6	RESERVED
412	(19C) CHARACTER	4	PDRDSSEQ	NEXT DATA SET RECORD SEQUENCE NUMBER

Problem Description Record Section

16	(10) STRUCTURE	0	PDRDEREC	PROBLEM DESCRIPTION RECORD
16	(10) CHARACTER	0	PDRDESCR	ARRAY OF PROBLEM DESCRIPTION LINES
16	(10) CHARACTER	72	PDRDESC	A LINE OF PROBLEM DESCRIPTION

OTB Data Record Section

16	(10)	STRUCTURE	0	PDRTRREC	OTB DATA RECORD FOR PROBLEM
16	(10)	CHARACTER	0	PDRTRARR	ARRAY OF OTB DATA LINES
16	(10)	CHARACTER	72	PDRTRKD	A LINE OF OTB DATA

HYMS Record Section

16	(10)	STRUCTURE	4	PDRHSREC	HYMS RECORD FOR PROBLEM
16	(10)	SIGNED	4	PDRHSLEN	LENGTH OF HYMS PROBLEM DESCRIPTION
20	(14)	CHARACTER	0	PDRHYMSD	HYMS PROBLEM DESCRIPTION

Data Set Record Section

16	(10)	STRUCTURE	64	PDRDSREC	DATA SET RECORD
16	(10)	CHARACTER	52	PDRDSNM	DATA SET NAME AND MEMBER
16	(10)	CHARACTER	44	PDRDSN	DATA SET NAME
60	(30)	CHARACTER	8	PDRMEMB	MEMBER NAME
68	(44)	CHARACTER	12	PDRDSRES	RESERVED

PDR Seed Record Section

16	(10)	STRUCTURE	8	PDRSVALU	PDR SEED VALUE
16	(10)	CHARACTER	3	PDRSVRES	RESERVED
19	(13)	CHARACTER	5	PDRSVNUM	PROBLEM NUMBER OF RECORD

BLSQEXTP

Common Name: BLSQEXTI Parameter List

Macro ID: BLSQEXTP

Created by: PRDMP, IPCS, SNAP

Pointed to by: The second parameter (a parameter list) that is passed to BLSQEXTI.

Function: Provides the BLSQEXTI with the information necessary to prepare for processing of dump processing exits under PRDMP, IPCS, and SNAP and to clean up when all such processing has completed.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	8	EXTP	BLSQEXTI input parameters
0	(0) BITSTRING 1... .. .1...11 1111	1	EXTPFUNC EXTPINIT EXTPDEL	Function code Function = initialize pointers Function = delete services reserved
1	(1) BITSTRING 1... .. .1...1...1 1111	1	EXTPSERV EXTPSFMT EXTPSECT EXTPSSEL EXTPSEQU	Service indicators AMDPREFMT, BLSQCFMT and BLSQIFMT BLSQECT and AMDPRECT BLSQSEL Equate and get symbol reserved
2	(2) CHARACTER	2		reserved
4	(4) ADDRESS	4	EXTPRETN	Address of returned load list

BLSQSMPL

Common Name: SUMMARY Parameter List

Macro ID: BLSQSMPL

Created by: BLSQSUM1, BLSRSUMM

Pointed to by: Parameter lists passed between modules of the common processor for the IPCS SUMMARY subcommand, the PRDMP SUMMARY control statement, and the PRDMP FORMAT control statement.

Function: Describe the options in effect for the processing of the current subcommand or control statement.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	88	SMPL	
0	(0) BITSTRING	1	SMPLFL1	Format control keyword flags
	1... ..		SMPLKF	KEYFIELD
	.1..		SMPLRG	REGISTERS
	..1.		SMPLTS	TCBSUMMARY
	...1		SMPLFM	FORMAT
	 1...		SMPLJS	JOBSUMMARY
	111			Reserved
1	(1) UNSIGNED	1	SMPLBLC	Blank line count
2	(2) BITSTRING	2	SMPLVCTL	View control overlay
4	(4) ADDRESS	4	SMPLSELO	-> Select service output
8	(8) BITSTRING	1	SMPLSELF	Select service flags
	1...		SMPLALL	All
	.1..		SMPLCUR	Current
	..1.		SMPLERR	Error
	...1		SMPLTCE	TCB error
	 1...		SMPLJL	JOBLIST
	1..		SMPLAS	ASIDLIST
	11			Reserved
9	(9) CHARACTER	3		Reserved
12	(C) ADDRESS	4	SMPLASLS	-> ASIDLIST PDE chain
16	(10) ADDRESS	4	SMPLJBLS	-> JOBNAME LIST PDE chain
20	(14) ADDRESS	4		Reserved
24	(18) ADDRESS	4	SMPLPFMT	-> PFMT for select report
28	(1C) ADDRESS	4	SMPLALST	Action list pointer
32	(20) ADDRESS	4	SMPLCBLD	Ctl block location in dump
36	(24) ADDRESS	4	SMPLCBLV	Ctl block location in virtual
40	(28) ADDRESS	4	SMPLLFPT	Last forward ptr compare
44	(2C) ADDRESS	4	SMPLGSPL	-> Global SVC priority list
48	(30) ADDRESS	4	SMPLGSMQ	-> Global SVC mgr Q
52	(34) ADDRESS	4	SMPLLSMQ	-> Local service mgr Q

BLSQSMPL - Storage Layout (continued)

<u>Offsets</u>	<u>Type</u>	<u>Length</u>	<u>Name</u>	<u>Description</u>
56	(38) ADDRESS	4	SMPLCSD	-> Common system data area
60	(3C) ADDRESS	4	SMPLPCCA	-> Physical CPU vector table
64	(40) ADDRESS	4	SMPLRTCT	-> Recovery/terminate ctl tbl
68	(44) ADDRESS	4		Reserved
72	(48) ADDRESS	4		Reserved
76	(4C) ADDRESS	4		Reserved
80	(50) ADDRESS	4		Reserved
84	(54) ADDRESS	4		Reserved

BLSQSRA

Common Name: Services Router Area

Macro ID: BLSQSRA

Created by: BLSRPRXC

Pointed to by: Field ADPLSRA (see "BLSABDPL" on page 326)

Function: Communicate between service routines used to provide common services to exits operating under PRDMP, IPCS, and SNAP.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	704	SRA	Services Router Area
0	(0) ADDRESS	4	SRACBATL	->CBAT load table
4	(4) ADDRESS	4	SRAECTP	->ECT (module AMDPRECT)
8	(8) UNSIGNED	2	SRAECTN	Number of ECT entries
10	(A) BITSTRING	1	SRAPFLGS	Processing flags
	1... ..		SRAPRNT	Communication between host and ECT service that the exit printed something
11	.111 1111 (B) CHARACTER	85	SRARSV1	reserved reserved
96	(60) ADDRESS	4	SRAALLOP	->BLSQALLO
100	(64) ADDRESS	4	SRAFREEP	->BLSQFREE
104	(68) CHARACTER	8	SRASTG	Stack control data
104	(68) ADDRESS	4	SRASTGP	->Current extent
108	(6C) UNSIGNED	4	SRASTGN	Bytes now available
112	(70) CHARACTER	256	SRATRHEX	TR table--BIT to CHAR
368	(170) CHARACTER	256		reserved

There are three inter-related definitions that pertain to structure SRAVECT:

1. Mapping macro BLSABDPL defines constants associated with the services addresses by its pointers. This macro provides the end-user with these values so that the values can be passed to ADPLESRV (module BLSQR0UT) as an indication of the service desired.
2. The following structured declaration of the vector is used for initialization purposes by module BLSQEXTI. It may also be employed for high-performance linkages between service routines.
3. Array SRASTAB provides a means for router BLSQR0UT to relate the constants passed by end-users to the address of the service routine in the table.

624	(270) CHARACTER	80	SRAVECT	Exit services table
-----	-----------------	----	---------	---------------------

BLSQSRA - Storage Layout (continued)

Offsets	Type	Length	Name	Description
624 (270)	ADDRESS	4	SRASACCP	->Storage access service
628 (274)	ADDRESS	4	SRASPRTP	->Print service
632 (278)	ADDRESS	4	SRASFMTP	->Format pattern service
636 (27C)	ADDRESS	4	SRASCBFP	->CB formatter service
640 (280)	ADDRESS	4	SRASNDXP	->Index service
644 (284)	ADDRESS	4	SRASECTP	->ECT exit service
648 (288)	ADDRESS	4	SRASSELP	->Select ASID service
652 (28C)	ADDRESS	4	SRASEQSP	->Equate symbol service
656 (290)	ADDRESS	4	SRASGTSP	->Get symbol service
660 (294)	ADDRESS	44		reserved

BLSRDATC

Common Name: Data Attribute Descriptor

Macro ID: BLSRDATC

Created by: IPCS subcommands concerned with debugging

Size: 60 bytes

Pointed to by: Parameter lists in which IPCS modules describe a data area, load module, subpool extent, ...

Function: Each data attribute descriptor summarizes the relocatable attributes of one block of storage, that is those attributes other than the address space and the address of the control block.

Note: The macro used to describe this structure permits the specification of the initial characters of each symbol. In this example, the characters DATC were chosen.

Storage Layout

<u>Offsets</u>	<u>Type</u>	<u>Length</u>	<u>Name</u>	<u>Description</u>
0	(0) STRUCTURE	60	DATC	BLSRDATC MF(L,DATC)
0	(0) CHARACTER	0	DATC00	BLSRDATC #MD33116

IPCS records the following properties for areas of storage:

- The offset between an addressed byte and the physical origin of this area.
- The physical length of this area.
- A data type.
- An indication whether the area is scalar or an array and, if the area is an array, the number of entries in the array and the subscript which applies to the first entry.

0	(0) SIGNED	4	DATCOF	Offset in bytes
4	(4) UNSIGNED	4	DATCLE	Length in bytes
8	(8) UNSIGNED	1	DATCOB	BLSRDATT MF(SUBLIST,DATCT,,2) BLSRDATT #MD83116 Data type code
9	(9) UNSIGNED	1	DATCLB	
10	(A) CHARACTER	34	DATCT	
10	(A) CHARACTER	0	DATCT0	
10	(A) CHARACTER	1	DATCTY	
11	(B) BITSTRING	1		
12	(C) CHARACTER	31	DATCTD	Data name
43	(2B) CHARACTER	1	DATCTE	reserved
44	(2C) CHARACTER	0	DATCT9	BLSRDATT #MD83116
44	(2C) UNSIGNED	4	DATCIM	Array entry count

BLSRDATC - Storage Layout (continued)

Offsets	Type	Length	Name	Description
48	(30) SIGNED	4	DATCIL	Subscript of initial array entry
52	(34) BITSTRING	4	DATCF DATCFA	Flags Array
	1....111 1111 1111 1111 1111 1111 1111 1111			
56	(38) BITSTRING	4		
60	(3C) CHARACTER	0	DATC99	BLSRDATC #MD83116

BLSRDATS

Common Name: Address Space Descriptor

Macro ID: BLSRDATS

Created by: IPCS subcommands concerned with debugging

Size: 16 bytes

Pointed to by: Parameter lists in which IPCS modules describe a data area, load module, subpool extent, ...

Function: Each address space descriptor defines one address space within which data might appear.

Note: The macro used to describe this structure permits the specification of the initial characters of each symbol. In this example, the characters DATS were chosen.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	16	DATS	BLSRDATS #MD83116
0	(0) CHARACTER	0	DATS0	BLSRDATS #MD83116

An IPCS address space is specified by three values:

1. A two-character code identifying the type of address space:
 - Letter code A indicates that the file contains an MVS high-speed dump and that absolute main storage of the dumped system is being referenced (ABSOLUTE)
 - Code BL indicates that a physical block in a file is being referenced using a relative block number (BLOCK)
 - Code BS indicates that a relative byte address group in the file is being referenced (RBA)
 - Code BT indicates that a physical block in a file is being referenced using a relative track address (TTR)
 - Letter code C indicates that the file contains an MVS high-speed dump and that the CPU status data for one dumped CPU is being referenced (STATUS)
 - Code CR indicates that the file contains an MVS high-speed dump and that real main storage seen by one CPU is being referenced (REAL)
 - Code CV indicates that the file contains an MVS high-speed dump and that virtual main storage seen by one MVS address space dispatched on a designated CPU is being referenced (ASID)
 - Letter code H indicates that the file contains an MVS high-speed dump and that the header data for the dump is being referenced (HEADER)
2. A binary integer whose interpretation depends on the preceding code:
 - For letter codes A and H this integer has no meaning and should be set to zero.
 - For code BL this integer should be the relative block number

- For code BS this integer should be the relative byte address group number
 - For code BT this integer should be the relative track address
 - For codes beginning with the letter C this integer contains the System/370 CPU address (STAP instruction) for the referenced CPU.
3. A binary integer whose interpretation depends on the preceding code:
- For code CV this integer contains the address space identification (ASID) for the referenced address space.
 - For other codes this integer has no meaning and should be set to zero.

0	(0) CHARACTER	2	DATST	Address space type code
2	(2) BITSTRING	2		
4	(4) UNSIGNED	4	DATS1	Integer 1
8	(8) UNSIGNED	4	DATS2	Integer 2
12	(C) BITSTRING	4		
16	(10) CHARACTER	0	DATS9	BLSRDATS #MD83116

BLSRDATT

Common Name: Data Type Descriptor

Macro ID: BLSRDATT

Created by: IPCS subcommands concerned with debugging

Size: 34 bytes

Pointed to by: Parameter lists in which IPCS modules describe a data area, load module, subpool extent, ...

Function: Each data type descriptor identifies the type of data stored in one block of storage.

Note: The macro used to describe this structure permits the specification of the initial characters of each symbol. In this example, the characters DATT were chosen.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	34	DATT	BLSRDATT MF(L,DATT)
0	(0) CHARACTER	0	DATT0	BLSRDATT #MD83116

IPCS records the data type in two parts:

1. A one-byte code similar to that employed by the System/370 assembler.
2. A Data name which refines the description of complex data types: AREA, MODULE, and STRUCTURE.

0	(0) CHARACTER	1	DATTY	Data type code
1	(1) BITSTRING	1		
2	(2) CHARACTER	31	DATTD	Data name
33	(21) CHARACTER	1	DATTE	reserved
34	(22) CHARACTER	0	DATT9	BLSRDATT #MD83116

BLSRDTDT

Common Name: Data Type Processing Table

Macro ID: BLSRDTDT

Created by: Assembly of BLSRDTM1 and BLSRDTU1

Size: 52 bytes

Pointed to by: Fields DUTSTRCP and DUTAREAP (see “BLSRDUT” on page 371)

Function: Provide information regarding complex data types for which IPCS provides special data area location or validation services. Two tables are actually present in IPCS:

- BLSRDTMT provides associations for structures (see DUTSTRCP)
- BLSRDTUT provides associations for areas (see DUTAREAP)

Both are associated with MVS/XA dumps.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	64	DTDT	
0	(0) CHARACTER	8	DTDTID	IDENTIFIER
8	(8) UNSIGNED	4	DTDTDIM	NUMBER OF DATA NAMES

Processing Definition Array

12	(C) CHARACTER	52	DTDTD	PROCESSING DEFINITION ARRAY
12	(C) UNSIGNED	1	DTDTDL	LENGTH OF DATA NAME
13	(D) CHARACTER	31	DTDTDTD	DATA NAME
44	(2C) UNSIGNED	4	DTDTDTN	INDEX OF NORMALIZED DATA NAME
48	(30) CHARACTER	8	DTDTDSEP	FIND EXIT ENTRY POINT NAME
56	(38) CHARACTER	8	DTDDEV	SCAN EXIT ENTRY POINT NAME

BLSRDUT

Common Name: Dump Table

Macro ID: BLSRDUT

Created by: Compilation of modules BLSRDUT1, BLSRDUT2, BLSRDUT3, BLSRDUT4, and BLSRDUT5

Pointed to by: Field ZZ6DUTP (see “BLSRZZ6” on page 400)

Function: Describe the special processing required for one type of dump supported by IPCS.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	256	DUT	DUMP TABLE
0	(0) CHARACTER	8	DUTID	TABLE IDENTIFIER
8	(8) CHARACTER	8	DUTCSECT	CONTROL SECTION IDENTIFIER
16	(10) CHARACTER	8	DUTDATE	DATE ASSEMBLED

Names for Dump Processing

24	(18) CHARACTER	8		RESERVED
32	(20) CHARACTER	8	DUTSEQ	SEQUENTIAL SETUP ROUTINE
40	(28) CHARACTER	8	DUTOPEN	DIRECT OPEN ROUTINE
48	(30) CHARACTER	8	DUTDIR	DIRECT SETUP ROUTINE
56	(38) CHARACTER	8	DUTDSL	MODULE DATA SEARCH ROUTINE
64	(40) CHARACTER	8	DUTDVL	MODULE DATA SCAN ROUTINE
72	(48) CHARACTER	96		RESERVED

Addresses for Dump Processing

168	(A8) ADDRESS	4	DUTSTRCP	->STRUCTURE TABLE
172	(AC) ADDRESS	4	DUTAREAP	->AREA TABLE
176	(B0) ADDRESS	4	DUTSSTP	->SYMBOLIC ALIAS TABLE
180	(B4) ADDRESS	52		RESERVED

Reserved Space

232	(E8) BITSTRING	24		RESERVED
-----	----------------	----	--	----------

BLSRDUT - Storage Layout (continued)

<u>Offsets</u>	<u>Type</u>	<u>Length</u>	<u>Name</u>	<u>Description</u>
256	(100) CHARACTER	0	DUT999999	END OF DUMP TABLE

BLSRESSY

Common Name: Dump Directory Equate Symbol Record

Macro ID: BLSRESSY

Created by: IPCS subcommands concerned with debugging

Size: 182 bytes + a remark containing 0-512 bytes of text

Pointed to by: Parameter lists in which IPCS modules describe a data area, load module, subpool extent,

Function: Each dump directory equate symbol record records the association between a symbol, e.g., CVT, ASCB001, ..., and the location and properties of a block of storage. IPCS modules also use this structure as a parameter to communicate the location of a block of storage.

Note: The macro used to describe this structure permits the specification of the initial characters of each symbol. In this example, the characters ESSY were chosen.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	694	ESSY	BLSRESSY BUFFER MF(L,ESSY)
0	(0) CHARACTER	2	ESSYRID	Record type==>ES
2	(2) BITSTRING	6		BLSURISY #MD82026
8	(8) CHARACTER	8		
16	(10) BITSTRING	8		

An Equate Symbol record defines one symbol, associating it with a contiguous block of storage in an address space.

24	(18) UNSIGNED	4	ESSYRDX	Data set index
28	(1C) CHARACTER	31	ESSYSYM	Equated symbol
59	(3B) CHARACTER	0	ESSYELK	End of logical key
59	(3B) CHARACTER	1	ESSYRV1	reserved
60	(3C) CHARACTER	16	ESSYAS	BLSRDATS #MD83116
60	(3C) CHARACTER	0	ESSYAS0	BLSRDATS #MD83116
60	(3C) CHARACTER	2	ESSYAST	Address space type code
62	(3E) BITSTRING	2		
64	(40) UNSIGNED	4	ESSYAS1	Integer 1
68	(44) UNSIGNED	4	ESSYAS2	Integer 2
72	(48) BITSTRING	4		
76	(4C) CHARACTER	0	ESSYAS9	BLSRDATS #MD83116
76	(4C) ADDRESS	4	ESSYLAD	Logical address

BLSRESSY - Storage Layout (continued)

Offsets	Type	Length	Name	Description
80	(50) CHARACTER	60	ESSYD	BLSRDATC MF(SUBLIST,ESSYD,,2)
80	(50) CHARACTER	0	ESSYD00	BLSRDATC #MD83116
80	(50) SIGNED	4	ESSYD0F	Offset in bytes
84	(54) UNSIGNED	4	ESSYDLE	Length in bytes
88	(58) UNSIGNED	1	ESSYD0B	BLSRDATT MF(SUBLIST,ESSYDT,,3) BLSRDATT #MD83116 Data type code
89	(59) UNSIGNED	1	ESSYDLB	
90	(5A) CHARACTER	34	ESSYDT	
90	(5A) CHARACTER	0	ESSYDT0	
90	(5A) CHARACTER	1	ESSYDTY	
91	(5B) BITSTRING	1		
92	(5C) CHARACTER	31	ESSYDTD	Data name
123	(7B) CHARACTER	1	ESSYDTE	reserved
124	(7C) CHARACTER	0	ESSYDT9	BLSRDATT #MD83116
124	(7C) UNSIGNED	4	ESSYDIM	Array entry count
128	(80) SIGNED	4	ESSYDIL	Subscript of initial array entry
132	(84) BITSTRING	4	ESSYDF ESSYDFA	Flags Array
	1... .. .111 1111 1111 1111 1111 1111 1111 1111			
136	(88) BITSTRING	4		
140	(8C) CHARACTER	0	ESSYD99	BLSRDATC #MD83116
140	(8C) ADDRESS	4	ESSYMAD	->1st byte missing
144	(90) BITSTRING	16		

System/370 storage key summary--the storage key(s) for the main storage described by this record are summarized by:

1. Taking the storage key for the first 2K block
2. Setting the reference and change bits in the key if they are on in any 2K block between the origin of the storage described by the symbol and the first storage not available in the dump.

160	(A0) BITSTRING	1	ESSYKEY	System/370 storage key. All bits on when indeterminate
-----	----------------	---	---------	--

Flags

161	(A1) BITSTRING	3	ESSYF	Flags
161	(A1) BITSTRING	1	ESSYFS	Storage flags
	1... ..		ESSYFSC	Storage information complete
	.1... ..		ESSYFS2	Multiple storage keys

BLSRESSY - Storage Layout (continued)

Offsets	Type	Length	Name	Description
	...1.		ESSYFSM	Not all storage in dump
	...1.		ESSYFSA	Absolute address, field ESSYABS, is valid
	 1...		ESSYFSP	Prefixed storage
	1..		ESSYFSR	Reclaimed storage
	1.		ESSYFSX	Multiple records
	1		ESSYFSS	SUMDUMP data
162	(A2) BITSTRING	1	ESSYFC	Control flags
	1...		ESSYFCD	Drop not permitted
	.111 1111			
163	(A3) BITSTRING	1	ESSYFC0	Common storage
	1...			
	.111 1111			
164	(A4) UNSIGNED	4	ESSYABS	Absolute address for this address
168	(A8) BITSTRING	12		

Remark--LENGTH(0:512) characters

180	(B4) CHARACTER	514	ESSYR	BLRPRM #MD82025
180	(B4) UNSIGNED	2	ESSYRL	Length of ESSYRT
182	(B6) CHARACTER	512	ESSYRT	BLRTEXT #MD82080
182	(B6) CHARACTER	512	ESSYRTA	Section 1
694	(2B6) CHARACTER	0	ESSY999	BLSRESSY #MD83116

BLSRFD

Common Name: Find Data

Macro ID: BLSRFD

Created by: BLSLFIND, BLSLFISE, BLSRFIND

Pointed to by: Parameter lists passed by the FIND primary command (BLSLFIND and BLSLFISE) and the FIND subcommand (BLSRFIND) to the FIND service routine, BLSRFISE.

Function: Describe the search criterion to be used by BLSRFISE when storage is to be searched.

Note: The macro used to describe this structure permits the specification of the initial characters of each symbol. In this example, the characters FD were chosen.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	536	FD	BLSRFD #MD83088
0	(0) CHARACTER	260	FDSE	BLSRVALY #MD83084
0	(0) CHARACTER	0	FDSE0	BLSRVALY #MD83084
0	(0) CHARACTER	1	FDSET	Data type code
1	(1) BITSTRING	1		
2	(2) UNSIGNED	2	FDSEL	Length of value (bytes)
4	(4) CHARACTER	256	FDSEC	EBCDIC value
4	(4) BITSTRING	256	FDSEX	Hexadecimal value
4	(4) SIGNED	4	FDSEF	Fullword numeric value
4	(4) SIGNED	2	FDSEH	Halfword numeric value
6	(6) SIGNED	2		
8	(8) BITSTRING	252		
260	(104) CHARACTER	0	FDSE9	BLSRVALY #MD83084

Mask Value

260	(104) CHARACTER	260	FDMA	BLSRVALY #MD83084
260	(104) CHARACTER	0	FDMA0	BLSRVALY #MD83084
260	(104) CHARACTER	1	FDMAT	Data type code
261	(105) BITSTRING	1		
262	(106) UNSIGNED	2	FDMAL	Length of value (bytes)
264	(108) CHARACTER	256	FDMAC	EBCDIC value
264	(108) BITSTRING	256	FDMAX	Hexadecimal value
264	(108) SIGNED	4	FDMAF	Fullword numeric value
264	(108) SIGNED	2	FDMAH	Halfword numeric value

BLSRFD - Storage Layout (continued)

Offsets	Type	Length	Name	Description
266	(10A) SIGNED	2		
268	(10C) BITSTRING	252		
520	(208) CHARACTER	0	FDMA9	BLSRVALY #MD83084

Boundary Alignment Required

520	(208) UNSIGNED	4	FDBDY	Boundary
524	(20C) UNSIGNED	4	FD0FF	Offset from boundary
528	(210) UNSIGNED	4		

Additional Search Options

532	(214) BITSTRING	3	FDF FDFBREAK	Search options Break option
	11111 1111 1111 1111 1111 1111			
535	(217) UNSIGNED	1	FDRELATE	Relationship sought
536	(218) CHARACTER	0	FD99999	BLSRFD #MD83088

BLSRHSD

Common Name: System/370 High Speed Dump Record Formats

Macro ID: BLSRHSD

Created by: AMDSADMP and SDUMP

Function: Record the state of System/370 CPU registers, main storage, and/or virtual storage at a point in time, usually at the time a problem has been detected.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	4104	HSDC	
0	(0) BITSTRING	1	HSDCTYPE	HSDTYPES
1	(1) BITSTRING	1	HSDCCLA	HSDCLAC
2	(2) BITSTRING	2	HSDCSF	FLAGS
1... ..			HSDCSFU	UNIPROCESSOR OR HALF-DUPLEX
.1... ..			HSDCSFI	STORE STATUS MAY BE INVALID
..1.			HSDCSFR	GENERAL PURPOSE REGS. VALID
...1 1111				
1111 1111				
4	(4) BITSTRING	2		
6	(6) UNSIGNED	2	HSDCSCPU	CPU ADDRESS

Image of the absolute prefixed storage area for the system after a store status operation for the indicated CPU

8	(8) CHARACTER	4096	HSDCPSA	PSA
8	(8) BITSTRING	256		
264	(108) BITSTRING	8	HSDCLPSW	STORE-STATUS PSW
272	(110) ADDRESS	4	HSDCPREF	->PSA (ABSOLUTE)
276	(114) BITSTRING	82		
358	(166) BITSTRING	2	HSDCADDR	CPU ADDRESS
360	(168) BITSTRING	32	HSDCFPR	FLOATING POINT REGS
392	(188) BITSTRING	64	HSDCGPR	GPRS
456	(1C8) BITSTRING	64	HSDCTLR	CONTROL REGISTERS
456	(1C8) BITSTRING	4	HSDC0	CONTROL REGISTER 0
1... ..			HSDC0SSM	SSM-SUPPRESSION
.1... ..			HSDC0TSM	TOD SYNCH MASK
..1.			HSDC0LAP	LOW ADDRESS PROTECTION
...1			HSDC0DUX	ASID EXTRACTION
.... 1....			HSDC0FEP	FETCH PROTECTION
.... .111				
11... ..			HSDC0ATF	TRANSLATION FORMAT
..1.			HSDC0IAM	MALFUNCTION ALERT MASK
...1			HSDC0ESM	EMERGENCY SIGNAL MASK
.... 1....			HSDC0ECM	EXTERNAL CALL MASK

BLSRHSD - Storage Layout (continued)

Offsets	Type	Length	Name	Description
.... .1..			HSDC0TSC	TOD SYNCH-CHECK MASK
.... ..1.			HSDC0CCM	CLOCK COMPARATOR MASK
.... ...1			HSDC0CTM	CPU TIMER MASK
1.... ..			HSDC0SES	SERVICE-SIGNAL MASK
.1..				RESERVED
..1.			HSDC0IKM	INTERRUPT-KEY MASK
...1 1111				
1.... ..				RESERVED
460 (1CC) BITSTRING		4	HSDC1	CONTROL REGISTER 1
1.... 1111			HSDC1SSE	SPACE-SWITCH-EVENT MASK
.111 1111				
1111 1111				
1111 1111			HSDC1STO	PRIMARY SEGTab ORIGIN
463 (1CF) UNSIGNED		1	HSDC1STL	PRIMARY SEGTab SIZE
464 (1D0) BITSTRING		4	HSDC2	CONTROL REGISTER 2
468 (1D4) BITSTRING		4	HSDC3	CONTROL REGISTER 3
468 (1D4) BITSTRING		2	HSDC3PKM	PSW-KEY MASK
470 (1D6) UNSIGNED		2	HSDC3ASN	SECONDARY ASN
472 (1D8) BITSTRING		4	HSDC4	CONTROL REGISTER 4
476 (1DC) BITSTRING		4	HSDC5	CONTROL REGISTER 5
1.... ..			HSDC5SLC	SUBSYSTEM-LINKAGE CONTROL
476 (1DC) BITSTRING		3	HSDC5LTO	LINKAGE-TABLE ORIGIN
.111 1111			HSDC5LTL	LINKAGE-TABLE SIZE
480 (1E0) BITSTRING		4	HSDC6	CONTROL REGISTER 6
480 (1E0) BITSTRING		1	HSDC6IIS	I/O-INTERRUPT-SUBCLASS
481 (1E1) BITSTRING		3		RESERVED
484 (1E4) BITSTRING		4	HSDC7	CONTROL REGISTER 7
1.... ..				RESERVED
.111 1111				
1111 1111				
1111 1111				
487 (1E7) UNSIGNED		1	HSDC7STO	SECONDARY SEGTab ORIGIN
			HSDC7STL	SECONDARY SEGTab SIZE
488 (1E8) BITSTRING		4	HSDC8	CONTROL REGISTER 8
488 (1E8) BITSTRING		2		RESERVED
490 (1EA) BITSTRING		2	HSDC8MOM	MONITOR MASKS
492 (1EC) BITSTRING		4	HSDC9	CONTROL REGISTER 9
1.... ..			HSDC9SBE	SUCCESSFUL-BRANCH EVENT
.1..			HSDC9IFE	INSTRUCTION-FETCH EVENT
..1.			HSDC9SAE	STORAGE-ALTERATION EVENT
...1			HSDC9GAE	GPR-ALTERATION EVENT
.... 1111				RESERVED
1111 1111				PER GPR MASKS
494 (1EE) BITSTRING		2	HSDC9GRM	
496 (1F0) BITSTRING		4	HSDCA	CONTROL REGISTER 10
496 (1F0) ADDRESS		4	HSDCAORG	PER STARTING ADDRESS
500 (1F4) BITSTRING		4	HSDCB	CONTROL REGISTER 11
500 (1F4) ADDRESS		4	HSDCBEND	PER ENDING ADDRESS

BLSRHSD - Storage Layout (continued)

Offsets	Type	Length	Name	Description
504	(1F8) BITSTRING	4	HSDCC HSDCCBTC	CONTROL REGISTER 12 BRANCH-TRACE CONTROL
	1.... ..			
	.1111 1111			
	1111 1111			
	1111 1111			
	1111 11..		HSDCCTEA	TRACE-ENTRY ADDRESS
	1		HSDCCATC	ASN-TRACE CONTROL
	1		HSDCCETC	EXPLICIT-TRACE CONTROL
508	(1FC) BITSTRING	4	HSDCD	CONTROL REGISTER 13
512	(200) BITSTRING	4	HSDCE	CONTROL REGISTER 14
	1.... ..			RESERVED
	.1..		HSDCESMC	SYNCHRONOUS-MCEL CONTROL
	..1.		HSDCECSD	CHANNEL-SUBSYSTEM-DAMAGE
	...1		HSDCERRM	RECOVERY-REPORT MASK
	 1...		HSDCEDRM	DEGRADATION-REPORT MASK
	1..		HSDCEEDR	EXTERNAL-DAMAGE-REPORT
	1.		HSDCEWAM	WARNING MASK
	1		HSDCEAMC	SYNCHRONOUS-MCEL CONTROL
	1....		HSDCEAFL	ASYNCHRONOUS-FIXED-LOG
	.11.			RESERVED
	...1		HSDCEATC	ASN-TRANSLATION CONTROL
	 1111			
	1111 1111			
	1111 111.		HSDCEATO	ASN-FIRST-TABLE ORIGIN
516	(204) BITSTRING	4	HSDCF	CONTROL REGISTER 15
516	(204) ADDRESS	4	HSDCFMCE	MCEL ADDRESS
520	(208) CHARACTER	3584		BODY OF CPU STATUS RECORD

BLSRPHSY

Common Name: Dump Directory Program History Record

Macro ID: BLSRPHSY

Created by: BLSR3270

Size: Variable 128 bytes to 3072 bytes

Function: Record historic information retained between executions of a program relative to a specific dump data set.

Note: The macro used to describe this structure permits the specification of the initial characters of each symbol. In this example, the characters PHSY were chosen.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	304	PHSY	BLSRPHSY #MD80227
0	(0) CHARACTER	2	PHSYRID	Record type==>PH
2	(2) BITSTRING	6		BLSURISY #MD82026
8	(8) CHARACTER	8		
16	(10) BITSTRING	8		

A program history record records contextual information saved between successive executions of a program and related to a particular dump data set

24	(18) UNSIGNED	4	PHSYRDX	Data set index
28	(1C) CHARACTER	8	PHSYPGM	Program name
36	(24) UNSIGNED	4	PHSYMRX	Multiple record index
40	(28) CHARACTER	0	PHSYELK	End of logical key
40	(28) BITSTRING	6		
46	(2E) CHARACTER	258	PHSYP	BLRPRM #MD82025
46	(2E) UNSIGNED	2	PHSYPL	Length of PHSYPT
48	(30) CHARACTER	256	PHSYPT	BLRTEXT #MD82080
48	(30) CHARACTER	256	PHSYPTA	Only section
304	(130) CHARACTER	0	PHSY999	BLSRPHSY #MD80227

BLSRPRX

Common Name: Exit Support Data

Macro ID: BLSRPRX

Created by: BLSRPRXC

Pointed to by: Field ZZ2PRXP (see "BLSUZZ2" on page 406)

Function: Communicate between service routines used to provide services to exits operating under IPCS.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	5904	PRX	Exit support data
0	(0) CHARACTER	3	PRXID	Identifier=>PRX
3	(3) UNSIGNED	1	PRXLEV	Modification level=>1
4	(4) CHARACTER	4		
8	(8) CHARACTER	256	PRX0BUF	Output buffer
264	(108) CHARACTER	4	PRXBR14	NOP Subroutine
268	(10C) CHARACTER	548		
816	(330) CHARACTER	16	PRXA	See PRXAS below
832	(340) CHARACTER	184	PRXES	ES record (macro BLSRESSY)
1016	(3F8) CHARACTER	64	PRXEXTN	ABDPL extension
1080	(438) CHARACTER	704	PRXSRA	Services router area (BLSQSRA)
1784	(6F8) CHARACTER	16	PRXCPPL	TSO CPPL (IKJCPPL)
1800	(708) CHARACTER	4104	PRXHSD	Dump record (macro BLSRHSD)
5904	(1710) CHARACTER	0	PRX99999	End of exit support data

ABDPL extension is pointed to by ADPTEXT.

1016	(3F8) STRUCTURE	64	ADPTEXTN	
1016	(3F8) ADDRESS	4	ADPLOPTR	->Operands buffer
1020	(3FC) ADDRESS	4	ADPLCPPL	->TSO CPPL
1024	(400) ADDRESS	4	ADPLRSV	reserved
1028	(404) UNSIGNED	2	ADPLCODE	Router return code
1030	(406) BITSTRING	1	ADPLPFLG	Processing flags
	1		ADPLNMSG	1-Suppress error messages
				reserved
1031	(407) BITSTRING	1	ADPLEFLG	Error flags
	1		ADPLEFAS	No automatic storage
				reserved

BLSPPRX - Storage Layout (continued)

Offsets	Type	Length	Name	Description
1032	(408) UNSIGNED	1	ADPLMAXL	Recommended maximum line width
1033	(409) UNSIGNED	1	ADPLSCOL	Control block formatting start column
1034	(40A) UNSIGNED	1	ADPLCOLS	Control block formatting column spacing
1035	(40B) CHARACTER	41	ADPLRSV1	reserved
816	(330) STRUCTURE	16	PRXAS	
816	(330) CHARACTER	0	PRXAS0	

An IPCS address space is specified by three values:

1. A two-character code identifying the type of address space:
 - Letter code A indicates that the file contains an MVS high-speed dump and that absolute main storage of the dumped system is being referenced (ABSOLUTE)
 - Code BL indicates that a physical block in a file is being referenced using a relative block number (BLOCK)
 - Code BS indicates that a relative byte address group in the file is being referenced (RBA)
 - Code BT indicates that a physical block in a file is being referenced using a relative track address (TTR)
 - Letter code C indicates that the file contains an MVS high-speed dump and that the CPU status data for one dumped CPU is being referenced (STATUS)
 - Code CR indicates that the file contains an MVS high-speed dump and that real main storage seen by one CPU is being referenced (REAL)
 - Code CV indicates that the file contains an MVS high-speed dump and that virtual main storage seen by one MVS address space dispatched on a designated CPU is being referenced (ASID)
 - Letter code H indicates that the file contains an MVS high-speed dump and that the header data for the dump is being referenced (HEADER)
2. A binary integer whose interpretation depends on the preceding code:
 - For letter codes A and H this integer has no meaning and should be set to zero.
 - For code BL this integer should be the relative block number
 - For code BS this integer should be the relative byte address group number
 - For code BT this integer should be the relative track address
 - For codes beginning with the letter C this integer contains the System/370 CPU address (STAP instruction) for the referenced CPU.
3. A binary integer whose interpretation depends on the preceding code:
 - For code CV this integer contains the address space identification (ASID) for the referenced address space.
 - For other codes this integer has no meaning and should be set to zero.

BLSRPRX - Storage Layout (continued)

<u>Offsets</u>	<u>Type</u>	<u>Length</u>	<u>Name</u>	<u>Description</u>
818	(332) BITSTRING	2		Integer 1
820	(334) UNSIGNED	4	PRXAS1	Integer 2
824	(338) UNSIGNED	4	PRXAS2	
828	(33C) BITSTRING	4		BLSRDATS #MD83116
832	(340) CHARACTER	0	PRXAS9	

BLSRRASY

Common Name: Dump Directory Address Space Record

Macro ID: BLSRRASY

Created by: BLSRDUIN, BLSRDUI2, BLSRDUI3, BLSRRACR, and BLSRRACV

Size: 148 bytes + 3-94 entries, each 32 bytes in length

Function: Record the non-trivial correspondence between MVS/XA absolute, real, virtual, CPU status, or header storage addresses and the dump data set records which contain the corresponding storage.

Note: The macro used to describe this structure permits the specification of the initial characters of each symbol. In this example, the characters RASY were chosen.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	3060	RASY	BLSRRASY #MD82080
0	(0) CHARACTER	2	RASYRID	Record type==>RA
2	(2) BITSTRING	6		BLSURISY #MD82026

An address space record provides the information necessary to retrieve data from an IPCS address space.

8	(8) UNSIGNED	4	RASYRDX	Data set index
12	(C) CHARACTER	16	RASYAS	BLSRDATS #MD83116
12	(C) CHARACTER	0	RASYAS0	BLSRDATS #MD83116
12	(C) CHARACTER	2	RASYAST	Address space type code
14	(E) BITSTRING	2		
16	(10) UNSIGNED	4	RASYAS1	Integer 1
20	(14) UNSIGNED	4	RASYAS2	Integer 2
24	(18) BITSTRING	4		
28	(1C) CHARACTER	0	RASYAS9	BLSRDATS #MD83116
28	(1C) ADDRESS	4	RASYFAD	->Final byte in the area
32	(20) CHARACTER	0	RASYELK	End of logical key
32	(20) BITSTRING	16		
48	(30) UNSIGNED	2	RASYDIM	Entries in RASYR array
50	(32) UNSIGNED	2	RASYUSE	Entries used in RASYR array

Array of entries, each of which describes one area within the referenced address space. The contents of an entire address space can be described by a single entry if

- The substring of each record containing part of the address space begins at the same record offset and has the same length
- The addresses of the records are of the form: M, M+1, M+2, M+3,

52	(34) CHARACTER	3008	RASYR	Address space description
52	(34) ADDRESS	4	RASYRIA	->Initial byte in the area
56	(38) UNSIGNED	4	RASYRLE	Length of data in each block
60	(3C) UNSIGNED	2		RESERVED
62	(3E) BITSTRING	2	RASYRF	Flags
	1... ..		RASYRFU	Storage not available
	.1.. ..		RASYRFP	Storage prefixed
	..1.		RASYRFR	Storage reclaimed
	...1		RASYRFT	Addresses translated
	 1...		RASYRFA	Absolute storage
	1..		RASYRFC	Common storage
	11			
	1111 1111			
64	(40) UNSIGNED	4	RASYRDA	1st block address
68	(44) UNSIGNED	4	RASYROF	Offset of data in each block
72	(48) UNSIGNED	4	RASYRNM	Number of blocks
76	(4C) BITSTRING	8		
3060	(BF4) CHARACTER	0	RASY999	BLSRRASY #MD82080

BLSRRBSY

Common Name: Dump Directory Block List Record

Macro ID: BLSRRBSY

Created by: Dynamically by module BLSRDUI3

Size: 128 bytes minumum, 3068 bytes maximum

Function: Correlates TTR addresses with relative block numbers for dump data sets in which records vary in length.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	3068	RBSY	BLSRRBSY #MD80192
0	(0) CHARACTER	2	RBSYRID	Record type==>RB
2	(2) BITSTRING	6		BLSURISY #MD82026

A block list record correlates block numbers with block addresses for a data set

8	(8) UNSIGNED	4	RBSYRDX	Data set index
12	(C) UNSIGNED	4	RBSYFBL	Final block described
16	(10) CHARACTER	0	RBSYELK	End of logical key
16	(10) BITSTRING	12		
28	(1C) UNSIGNED	2	RBSYDIM	Number of entries present in the RBSYR array
30	(1E) UNSIGNED	2	RBSYUSE	Number of entries used in the RBSYR array

Array of entries, each of which describes one group of blocks within which the addresses form an arithmetic sequence: n, n+1, n+2, ... and within which all blocks have the same length

32	(20) CHARACTER	3036	RBSYR	Block list description
32	(20) UNSIGNED	4	RBSYRBL	Initial block number
36	(24) UNSIGNED	2	RBSYRLE	Length of each block
38	(26) UNSIGNED	2	RBSYRNM	Number of blocks
40	(28) UNSIGNED	4	RBSYRDA	First TTR for group
3068	(BFC) CHARACTER	0	RBSY999	BLSRRBSY #MD80192

BLSRRCSY

Common Name: Dump Directory Contents List Record

Macro ID: BLSRRCSY

Created by: Dynamically by module BLSRDUI3

Size: 128 bytes mininum, 3072 bytes maximum

Function: Enumerates TTR addresses for dump data sets in which the records vary in length.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	3072	RCSY	BLSRRCSY #MD80192
0	(0) CHARACTER	2	RCSYRID	Record type==>RC
2	(2) BITSTRING	6		BLSURISY #MD82026

A contents list record lists block addresses within a data set

8	(8) UNSIGNED	4	RCSYRDX	Data set index
12	(C) UNSIGNED	4	RCSYFBL	Final block described
16	(10) CHARACTER	0	RCSYELK	End of logical key
16	(10) BITSTRING	12		
28	(1C) UNSIGNED	2	RCSYDIM	Number of entries present in the RCSYR array
30	(1E) UNSIGNED	2	RCSYUSE	Number of entries used in the RCSYR array

Array of entries, each of which describes one group of blocks within which the block addresses form an arithmetic sequence: n n+1, n+2, ... and within which all blocks have the same length

32	(20) CHARACTER	3040	RCSYR	Contents list description
32	(20) UNSIGNED	4	RCSYRBL	Initial block address
36	(24) UNSIGNED	2	RCSYRLE	Length of each block
38	(26) UNSIGNED	2	RCSYRNM	Number of blocks
3072	(C00) CHARACTER	0	RCSY999	BLSRRCSY #MD80192

BLSRRDSY

Common Name: Dump Directory Data Set Record

Macro ID: BLSRRDSY

Created by: BLSRDUIN

Function: Record the correspondence between the name of a cataloged data set and the numeric index used by IPCS dump data access modules to reference the data set. Also record key information pertaining to management of the data set.

Note: The macro used to describe this structure permits the specification of the initial characters of each symbol. In this example, the characters RDSY were chosen.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	128	RDSY	BLSRRDSY MF(L,RDSY)
0	(0) CHARACTER	2	RDSYRID	Record type==>RD
2	(2) BITSTRING	6		BLSURISY #MD82026

A DSNNAME record associates the name of a catalogued data set with an index value. The key of a DSNNAME record permits IPCS to recognize when records related to the data set have been saved across sessions.

Data Set Name

8	(8) CHARACTER	48	RDSYD	Variable-length record
8	(8) UNSIGNED	2	RDSYDL	Record length
10	(A) UNSIGNED	2	RDSYDO	Offset
12	(C) CHARACTER	44	RDSYDT	BLRTEXT #MD82030
12	(C) CHARACTER	44	RDSYDTA	Section 1

Member Name

56	(38) CHARACTER	8	RDSYMEM	Member name
64	(40) CHARACTER	0	RDSYELK	End of logical key
64	(40) UNSIGNED	4	RDSYRDX	Data set index

Preferred Address Space for this Data Set

68	(44) CHARACTER	16	RDSYQA	BLSRDATS #MD83116
68	(44) CHARACTER	0	RDSYQA0	BLSRDATS #MD83116

BLSRRDSY - Storage Layout (continued)

Offsets	Type	Length	Name	Description
68	(44) CHARACTER	2	RDSYQAT	Address space type code
70	(46) BITSTRING	2		
72	(48) UNSIGNED	4	RDSYQA1	Integer 1
76	(4C) UNSIGNED	4	RDSYQA2	Integer 2
80	(50) BITSTRING	4		
84	(54) CHARACTER	0	RDSYQA9	BLSRDATS #MD83116
84	(54) UNSIGNED	4	RDSYQ1	Active CPU address
88	(58) UNSIGNED	4	RDSYQ2	Active ASID
92	(5C) BITSTRING	4	RDSYF	Flags

Dump Characteristics

1... ..	RDSYFBS	RECFM=F
.1... ..	RDSYFDU	MVS dump
..1... ..	RDSYFMP	Multi-processor dump
...1... ..	RDSYFAB	Absolute storage
.... 1...	RDSYFST	CPU status data
.... .1...	RDSYFSU	Summary dump data
.... ..11		

Setup Status

1... ..	RDSYFIS	Need sequential setup
.1... ..	RDSYFID	Need direct setup
..1... ..	RDSYFPH	BLSRDUAL created PH record
...1 1111		
94 (5E) BITSTRING	1	
95 (5F) UNSIGNED	1	RDSYFNV
		Number of volumes
96 (60) ADDRESS	4	
100 (64) UNSIGNED	4	RDSYDA1
		First disk record address not yet read
104 (68) UNSIGNED	4	RDSYDA2
		Low disk record address beyond data set
108 (6C) ADDRESS	4	RDSYPXP
		->PRIVATEX (MVS dumps)
112 (70) ADDRESS	4	RDSYCAP
		->Common area (MVS dumps)
116 (74) UNSIGNED	4	RDSYBLK
		Blocks in data set
120 (78) BITSTRING	8	RDSYBYT
		Bytes in data set (packed decimal)
128 (80) CHARACTER	0	RDSY999
		BLSRRDSY #MD83118

BLSRSASY

Common Name: Dump Directory Storage Address Record

Macro ID: BLSRSASY

Created by: IPCS subcommands concerned with debugging

Size: 140 bytes + scan results using 0-2932 bytes

Function: Record the existence of a requested block of storage. Permit IPCS to recognize the extent to which analysis of the storage has been completed and unusual circumstances have been diagnosed.

Note: The macro used to describe this structure permits the specification of the initial characters of each symbol. In this example, the characters SASY were chosen.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	256	SASY	BLSRSASY MF(L,SASY)
0	(0) CHARACTER	2	SASYRID	Record type==>SA
2	(2) BITSTRING	6		BLSURISY #MD82026

A Storage address record identifier a block known to IPCS

8	(8) UNSIGNED	4	SASYRDX	Data set index
12	(C) CHARACTER	16	SASYAS	BLSRDATS #MD83116
12	(C) CHARACTER	0	SASYAS0	BLSRDATS #MD83116
12	(C) CHARACTER	2	SASYAST	Address space type code
14	(E) BITSTRING	2		
16	(10) UNSIGNED	4	SASYAS1	Integer 1
20	(14) UNSIGNED	4	SASYAS2	Integer 2
24	(18) BITSTRING	4		
28	(1C) CHARACTER	0	SASYAS9	BLSRDATS #MD83116
28	(1C) ADDRESS	4	SASYLAD	Logical address of the area
32	(20) CHARACTER	34	SASYDT	BLSRDATT MF(SUBLIST,SASYDT,,2)
32	(20) CHARACTER	0	SASYDT0	BLSRDATT #MD83116
32	(20) CHARACTER	1	SASYDTY	Data type code
33	(21) BITSTRING	1		
34	(22) CHARACTER	31	SASYDTD	Data name
65	(41) CHARACTER	1	SASYDTE	reserved
66	(42) CHARACTER	0	SASYDT9	BLSRDATT #MD83116
66	(42) CHARACTER	0	SASYELK	End of logical key
66	(42) BITSTRING	2		
68	(44) CHARACTER	60	SASYF	BLSRDATC MF(SUBLIST,SASYF,,2)

BLSRSASY - Storage Layout (continued)

Offsets	Type	Length	Name	Description
68	(44) CHARACTER	0	SASYF00	BLSRDATA #MD83116
68	(44) SIGNED	4	SASYF0F	Offset in bytes
72	(48) UNSIGNED	4	SASYFLE	Length in bytes
76	(4C) UNSIGNED	1	SASYF0B	BLSRDATT MF(SUBLIST,SASYFT,,3) BLSRDATT #MD83116 Data type code
77	(4D) UNSIGNED	1	SASYFLB	
78	(4E) CHARACTER	34	SASYFT	
78	(4E) CHARACTER	0	SASYFT0	
78	(4E) CHARACTER	1	SASYFTY	
79	(4F) BITSTRING	1		
80	(50) CHARACTER	31	SASYFTD	Data name
111	(6F) CHARACTER	1	SASYFTE	reserved
112	(70) CHARACTER	0	SASYFT9	BLSRDATT #MD83116
112	(70) UNSIGNED	4	SASYFIM	Array entry count
116	(74) SIGNED	4	SASYFIL	Subscript of initial array entry
120	(78) BITSTRING	4	SASYFF SASYFFA	Flags Array
	1... .. .111 1111 1111 1111 1111 1111 1111 1111			
124	(7C) BITSTRING	4		
128	(80) CHARACTER	0	SASYF99	BLSRDATA #MD83116

Scan results--Flags, return code, and processing summary

128	(80) BITSTRING	8	SASYSF SASYSF1 SASYSF9 SASYSFI SASYSFS SASYSFA	Scan flags Scan started Scan completed Initial analysis error Storage required for diagnostic(s) Storage attributes unknown
	1... .. .1...1...1... 1...111			
129	(81) BITSTRING	7		
136	(88) BITSTRING	8	SASYGMT	TOD clock value when scan was last performed
144	(90) CHARACTER	8	SASYPGV	Scan program name
152	(98) UNSIGNED	1	SASYSRC	Scan result code--0=>Informational, 4=>Warning, 8=>Error, 12=>Serious
153	(99) BITSTRING	101		

Scan processing details--LENGTH(0:2816) characters--structured by each scan exit to match its unique requirements

254	(FE) CHARACTER	2	SASYC	BLRPRM #MD82025
254	(FE) UNSIGNED	2	SASYCL	Length of SASYCT

BLSRSASY - Storage Layout (continued)

<u>Offsets</u>	<u>Type</u>	<u>Length</u>	<u>Name</u>	<u>Description</u>
256	(100) CHARACTER	0	SASYCT	BLRTEXT #MD82080
256	(100) CHARACTER	0	SASYCTA	Only section
256	(100) CHARACTER	0	SASY999	BLSRSASY #MD82026

BLSRSDSY

Common Name: Dump Directory Session Defaults

Macro ID: BLSRSDSY

Created by: FIND (BLSRFIND) and SETDEF (BLSRSETD) subcommand processors.

Size: 128 bytes minimum, 3072 bytes maximum

Function: Contains defaults established by the user.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	304	SDSY	BLSRSDSY #MD80196
0	(0) CHARACTER	2	SDSYRID	Record type==>SD
2	(2) BITSTRING	6		BLSURISY #MD82026
8	(8) CHARACTER	8		
16	(10) BITSTRING	8		

A session defaults record records IPCS session defaults saved between successive IPCS sessions

24	(18) CHARACTER	8	SDSYPGM	Program name
32	(20) UNSIGNED	4	SDSYMRX	Multiple record index
36	(24) CHARACTER	0	SDSYELK	End of logical key
36	(24) BITSTRING	10		
46	(2E) CHARACTER	258	SDSYP	BLRPRM #MD82025
46	(2E) UNSIGNED	2	SDSYPL	Length of SDSYPT
48	(30) CHARACTER	256	SDSYPT	BLRTXT #MD82080
48	(30) CHARACTER	256	SDSYPTA	Only section
304	(130) CHARACTER	0	SDSY999	BLSRSDSY #MD80196

BLSRSST

Common Name: Special Symbol Table

Macro ID: BLSRSST

Created by: Compilation of module BLSRSST1

Pointed to by: Field DUTSSTP (see “BLSRDUT” on page 371)

Function: Defines special symbols for MVS/XA dumps

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	28	SST	Special symbol table
0	(0) CHARACTER	3	SSTID	Table identifier
3	(3) UNSIGNED	1	SSTVER	Version of table
4	(4) CHARACTER	8	SSTCSECT	CSECT identifier
12	(C) CHARACTER	8	SSTLEVEL	Level identifier
20	(14) UNSIGNED	4	SSTDIM	Number of symbols defined

Special symbol definition array - Sorted by prefix

24	(18) CHARACTER	4	SSTS	Symbol definition array
24	(18) UNSIGNED	1	SSTST	Type of suffix
25	(19) UNSIGNED	1	SSTSLEN	Length of prefix
26	(1A) ADDRESS	2	SSTSOFF	Offset of prefix

BLSRVALY

Common Name: Task Variable

Macro ID: BLSRVALY

Created by: BLSLFISE, BLSRVAL

Pointed to by: Parameter lists passed to BLSRVAL

Function: Describe a general value used by the IPCS COMPARE and FIND subcommands and the FIND primary commands for the BLSLDISP dialog and the dump display reporter.

Note: The macro used to describe this structure permits the specification of the initial characters of each symbol. In this example, the characters VALY were chosen.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	260	VALY	BLSRVALY #MD83084
0	(0) CHARACTER	0	VALY0	BLSRVALY #MD83084

A value data buffer records a data type, data length, and a binary value

0	(0) CHARACTER	1	VALYT	Data type code
1	(1) BITSTRING	1		
2	(2) UNSIGNED	2	VALYL	Length of value (bytes)
4	(4) CHARACTER	256	VALYC	EBCDIC value
4	(4) BITSTRING	256	VALYX	Hexadecimal value
4	(4) SIGNED	4	VALYF	Fullword numeric value
4	(4) SIGNED	2	VALYH	Halfword numeric value
6	(6) SIGNED	2		
8	(8) BITSTRING	252		
260	(104) CHARACTER	0	VALY9	BLSRVALY #MD83084

BLSRVT

Common Name: Debugging Vector

Macro ID: BLSRVT

Created by: Compilation of module BLSRVECT

Pointed to by: Field ZZ2RVTP (see “BLSUZZ1” on page 404)

Function: Provide the addresses of common subroutines and data shared by the IPCS debugging subcomponents.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	384	RVT	BLRVCT #MD83078
0	(0) ADDRESS	4	RVTADDRP	->BLSRADDR
4	(4) ADDRESS	56	RVT0002P	reserved
60	(3C) ADDRESS	4	RVTACCLP	->BLSRACCL
64	(40) ADDRESS	4	RVTACCP	->BLSRACC
68	(44) ADDRESS	12	RVT0018P	reserved
80	(50) ADDRESS	4	RVTESPUP	->BLSRESPU
84	(54) ADDRESS	4	RVTESGEP	->BLSRESGE
88	(58) ADDRESS	4	RVTESGUP	->BLSRESGU
92	(5C) ADDRESS	4	RVTESARP	->BLSRESAR
96	(60) ADDRESS	4	RVTRAARP	->BLSRRAAR
100	(64) ADDRESS	4	RVTRAGEP	->BLSRRAGE
104	(68) ADDRESS	4	RVTADDP	->BLSRADDP
108	(6C) ADDRESS	4	RVTACCQP	->BLSRACCQ
112	(70) ADDRESS	12	RVT0029P	reserved
124	(7C) ADDRESS	4	RVTSAGUP	->BLSRSAGU
128	(80) ADDRESS	4	RVTSAGEP	->BLSRSAGE
132	(84) ADDRESS	4	RVTESCKP	->BLSRESCK
136	(88) ADDRESS	4	RVTMSGAP	->BLSRMSGA
140	(8C) ADDRESS	4	RVTMSGDP	->BLSRMSGD
144	(90) ADDRESS	16	RVT0037P	reserved
160	(A0) ADDRESS	4	RVTADDTP	->BLSRADDT
164	(A4) ADDRESS	4	RVTADDUP	->BLSRADDU
168	(A8) ADDRESS	4	RVTDUSEP	->BLSRDUSE

BLSRVT - Storage Layout (continued)

Offsets	Type	Length	Name	Description
172	(AC) ADDRESS	4	RVTDUSOP	->BLSRDUSO
176	(B0) ADDRESS	4	RVTADD1P	->BLSRADD1
180	(B4) ADDRESS	4	RVTADD2P	->BLSRADD2
184	(B8) ADDRESS	4	RVTSAGP	->BLSRSAG
188	(BC) ADDRESS	4	RVTMSGBP	->BLSRMSGB
192	(C0) ADDRESS	4	RVTADDSP	->BLSRADD5
196	(C4) ADDRESS	4	RVTADD3P	->BLSRADD3
200	(C8) ADDRESS	4	RVT0051P	reserved
204	(CC) ADDRESS	4	RVTPADSP	->BLSRPADS
208	(D0) ADDRESS	24	RVT0053P	reserved
232	(E8) ADDRESS	4	RVTADD4P	->BLSRADD4
236	(EC) ADDRESS	4	RVTSAPCP	->BLSRSAPC
240	(F0) ADDRESS	4	RVTSARP	->BLSRSAAR
244	(F4) ADDRESS	4	RVTSAGNP	->BLSRSAGN
248	(F8) ADDRESS	4	RVT0063P	reserved
252	(FC) ADDRESS	4	RVTSAGCP	->BLSRSAGC
256	(100) ADDRESS	4	RVTSAGQP	->BLSRSAGQ
260	(104) ADDRESS	4	RVTESGCP	->BLSRESGC
264	(108) ADDRESS	4	RVTESGQP	->BLSRESGQ
268	(10C) ADDRESS	4	RVTZALCP	->BLSRZALC
272	(110) ADDRESS	4	RVTZALNP	->BLSRZALN
276	(114) ADDRESS	4	RVTDUCKP	->BLSRDUCK
280	(118) ADDRESS	4	RVTDUCCP	->BLSRDUCC
284	(11C) ADDRESS	4	RVTSSSCP	->BLSRSSSC
288	(120) ADDRESS	4	RVTSSSPP	->BLSRSSSP
292	(124) ADDRESS	4	RVTSAGPP	->BLSRSAGP
296	(128) ADDRESS	4	RVTSAGRP	->BLSRSAGR
300	(12C) ADDRESS	4	RVTSAGSP	->BLSRSAGS
304	(130) ADDRESS	4	RVTDATKP	->BLSRDATK
308	(134) ADDRESS	4	RVTESSAP	->BLSRESSA
312	(138) ADDRESS	4	RVTESSRP	->BLSRESSR
316	(13C) ADDRESS	4	RVTDUCDP	->BLSRDUCD
320	(140) ADDRESS	4	RVTDUCLP	->BLSRDUCL

BLSRVT - Storage Layout (continued)

Offsets	Type	Length	Name	Description
324 (144)	ADDRESS	4	RVTACCBP	->BLSRACCB
328 (148)	ADDRESS	4	RVDUBFP	->BLSRDUBF
332 (14C)	ADDRESS	4	RVTRBGEP	->BLSRRBGE
336 (150)	ADDRESS	4	RVTRCGEP	->BLSRRCGE
340 (154)	ADDRESS	4	RVTM300P	->BLSRM300
344 (158)	ADDRESS	4	RVTM301P	->BLSRM301
348 (15C)	ADDRESS	4	RVTM302P	->BLSRM302
352 (160)	ADDRESS	4	RVTM077P	->BLSRM077
356 (164)	ADDRESS	4	RVTSSSAP	->BLSRSSSA
360 (168)	ADDRESS	4	RVTESGAP	->BLSRESGA
364 (16C)	ADDRESS	4	RVTACCNP	->BLSRACCN
368 (170)	ADDRESS	4	RVTMVSAP	->BLSRMVSA
372 (174)	ADDRESS	4	RVTMDSNP	->BLSRMDSN
376 (178)	ADDRESS	4	RVTADGFP	->BLSRADGP
380 (17C)	ADDRESS	4	RVTMVCLP	->BLSRMVCL
384 (180)	CHARACTER	0	RVT99999	BLRVCT #11D83078

BLSRZZ6

Common Name: Dump File Control Block

Macro ID: BLSRZZ6

Pointed to by: Fields ZZ2AZZ6P and ZZ2ZZ6P (see “BLSUZZ2” on page 406) and field ZZ6ZZ6P (see “BLSRZZ6”)

Function: Identify a dump data set that is open for processing by IPCS.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	2048	ZZ6	BLSRZZ6 #MD83121
0	(0) CHARACTER	0	ZZ60000	BLSRZZ6 #MD83121

Data Control Block

0	(0) BITSTRING	128	ZZ6RDCB	DCB
---	---------------	-----	---------	-----

DSNAME Record

128	(80) CHARACTER	128	ZZ6D	BLSRRDSY MF(SUBLIST,ZZ6D,,2)
128	(80) CHARACTER	2	ZZ6DRID	Record type==>RD
130	(82) BITSTRING	6		BLSURISY #MD82026
136	(88) CHARACTER	48	ZZ6DD	Variable-length record
136	(88) UNSIGNED	2	ZZ6DDL	Record length
138	(8A) UNSIGNED	2	ZZ6DD0	Offset
140	(8C) CHARACTER	44	ZZ6DDT	BLRXTX #MD82080
140	(8C) CHARACTER	44	ZZ6DDTA	Section 1
184	(B8) CHARACTER	8	ZZ6DI1EM	Member name
192	(C0) CHARACTER	0	ZZ6DELK	End of logical key
192	(C0) UNSIGNED	4	ZZ6DRDX	Data set index
196	(C4) CHARACTER	16	ZZ6DQA	BLSRDATS #MD83116
196	(C4) CHARACTER	0	ZZ6DQA0	BLSRDATS #MD83116
196	(C4) CHARACTER	2	ZZ6DQAT	Address space type code
198	(C6) BITSTRING	2		
200	(C8) UNSIGNED	4	ZZ6DQA1	Integer 1
204	(CC) UNSIGNED	4	ZZ6DQA2	Integer 2
208	(D0) BITSTRING	4		

BLSRZZ6 - Storage Layout (continued)

Offsets	Type	Length	Name	Description
212	(D4) CHARACTER	0	ZZ6DQA9	BLSRDATS #MD83116
212	(D4) UNSIGNED	4	ZZ6DQ1	Active CPU address
216	(D8) UNSIGNED	4	ZZ6DQ2	Active ASID
220	(DC) BITSTRING	4	ZZ6DF	Flags
	1... ..		ZZ6DFBS	RECFM=F
	.1... ..		ZZ6DFDU	MVS dump
	..1.		ZZ6DFMP	Multi-processor dump
	...1		ZZ6DFAB	Absolute storage
	 1...		ZZ6DFST	CPU status data
	1..		ZZ6DFSU	Summary dump data
	11			
	1... ..		ZZ6DFIS	Need sequential setup
	.1... ..		ZZ6DFID	Need direct setup
	..1.		ZZ6DFPH	BLSRDUAL created PH record
	...1 1111			
222	(DE) BITSTRING	1		
223	(DF) UNSIGNED	1	ZZ6DFNV	Number of volumes
224	(E0) ADDRESS	4		
228	(E4) UNSIGNED	4	ZZ6DDA1	First disk record address not yet read
232	(E8) UNSIGNED	4	ZZ6DDA2	Low disk record address beyond data set
236	(EC) ADDRESS	4	ZZ6DPXP	->PRIVATEX (MVS dumps)
240	(F0) ADDRESS	4	ZZ6DCAP	->Common area (MVS dumps)
244	(F4) UNSIGNED	4	ZZ6DBLK	Blocks in data set
248	(F8) BITSTRING	8	ZZ6DBYT	Bytes in data set (packed decimal)
256	(100) CHARACTER	0	ZZ6D999	BLSRRDSY #MD83118

Data Set Control Information

256	(100) BITSTRING	140		
396	(18C) CHARACTER	8	ZZ6DUTNM	Dump table name
404	(194) ADDRESS	4	ZZ6DUTP	->Dump table
408	(198) BITSTRING	96	ZZ6DSCB	DSCB--Data portion
504	(1F8) BITSTRING	8	ZZ6F	Flags
	1... ..		ZZ6FTAPE	Dump on tape
	.1... ..		ZZ6FMVOL	Multi-volume dump
	..1.		ZZ6FOPEN	Explicit OPEN request
	...1 1111			
	1111 1111			
	1111 1111			
	1111 1111			
	1111 1111			
	1111 1111			
	1111 1111			
	1111 1111			
	1111 1111			
512	(200) ADDRESS	4	ZZ6ZZ6P	Chain address

BLSRZZ6 - Storage Layout (continued)

Offsets	Type	Length	Name	Description
516 (204)	UNSIGNED	2	ZZ6BUFN	Number of buffers allocated
518 (206)	UNSIGNED	2	ZZ6BUFL	Length of each buffer allocated
520 (208)	UNSIGNED	4	ZZ6BUFB	Requests for data (excluding initial scan)
524 (20C)	CHARACTER	8	ZZ6FILE	File name
532 (214)	ADDRESS	4	ZZ6BUFP	->Array of buffers
536 (218)	UNSIGNED	4	ZZ6BUFI	Direct reads
540 (21C)	ADDRESS	4	ZZ6ACCXP	->Direct read subroutine
544 (220)	UNSIGNED	2	ZZ6BUFU	Number of buffers in use
546 (222)	UNSIGNED	2	ZZ6BUFJ	Index of last buffer used
548 (224)	CHARACTER	20	ZZ6V	DEVTYPE data
548 (224)	BITSTRING	4	ZZ6VE	UCBTYP
548 (224)	BITSTRING	2		
550 (226)	BITSTRING	1	ZZ6VETAP	Magnetic tape
	1... ..			
	.1.. ..		ZZ6VEDAS	Direct access storage
	..1.			
	...1 1111			
551 (227)	BITSTRING	1	ZZ6VDCTE	Device characteristics table entry
552 (228)	UNSIGNED	4	ZZ6VBLKL	Maximum BLKSIZE
556 (22C)	BITSTRING	8		
564 (234)	UNSIGNED	1	ZZ6VADJU	BLKSIZE adjustment
565 (235)	BITSTRING	3		
568 (238)	UNSIGNED	4	ZZ6TRCAP	Maximum-length blocks per DASD track
572 (23C)	BITSTRING	12		
584 (248)	CHARACTER	8	ZZ6MV	Current volume data
584 (248)	UNSIGNED	4	ZZ6MVORG	Base record--this volume
588 (24C)	UNSIGNED	4	ZZ6MVEND	Base record--next volume
592 (250)	BITSTRING	176	ZZ6JFCB	JFCB (macro IEFJFCBN)

Buffer Management Data Array, One Entry for Each Buffer

768 (300)	CHARACTER	1280	ZZ6B	Buffer management data array
768 (300)	UNSIGNED	4	ZZ6BDA	Disk record address
772 (304)	UNSIGNED	2	ZZ6BFL	Length of buffer contents
774 (306)	BITSTRING	2		
776 (308)	UNSIGNED	4	ZZ6BRC	References to this buffer since it was filled with its current contents
780 (30C)	UNSIGNED	4	ZZ6BRB	References to data set when this buffer was filled with its current contents
2048 (800)	CHARACTER	0	ZZ69999	BLSRZZ6 #MD83121

BLSUVSFB

Common Name: Dump Directory Control Block

Macro ID: BLSUVSFB

Pointed to by: Field ZZ1ACBP (see “BLSUZZ1” on page 404)

Function: Store information required for the processing of the dump directory.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	512	ZZ4	BLSUVSFB #MD81313
0	(0) CHARACTER	256	ZZ4ACB	ACB
256	(100) CHARACTER	8	ZZ4ID	Identifier
264	(108) BITSTRING	2		

Data Set Name

266	(10A) CHARACTER	46	ZZ4D	BLRPRM #MD82025
266	(10A) UNSIGNED	2	ZZ4DL	Length of ZZ4DT
268	(10C) CHARACTER	44	ZZ4DT	BLRTXT #MD82030
268	(10C) CHARACTER	44	ZZ4DTA	Section 1
312	(138) ADDRESS	4	ZZ4P	->Next file control block.
316	(13C) ADDRESS	4	ZZ4RLP	->Record list
320	(140) BITSTRING	192		
512	(200) CHARACTER	0	ZZ49999	BLSUVSFB #MD81313

BLSUZZ1

Common Name: Common Area

Macro ID: BLSUZZ1

Created by:

- Compilation of module BLSUPUT
- Dynamically by module BLS9CMD1

Pointed to by: Field ZZZZZ1P (see “BLSUZZ2” on page 406)

Function: Contain information shared by all tasks in the IPCS environment.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	512	ZZ1	BLSUZZ1 #MD82236
0	(0) CHARACTER	4	ZZ1IDE	Literal identifier
4	(4) ADDRESS	4	ZZ1NULL	Null pointer value for IPCS
8	(8) ADDRESS	4	ZZ1PNULL	IKJPARS PDE null pointer value
12	(C) ADDRESS	4	ZZ1VRPP	->BLDVPR/DLVRP parameter list
16	(10) ADDRESS	4		
20	(14) ADDRESS	4	ZZ1PDCBP	->PRD (macro BLSUPRD)
24	(18) ADDRESS	4	ZZ1TDCBP	->tasklib DCB
28	(1C) ADDRESS	4	ZZ1ACBP	->FILE(IPCSDDIR) control block
32	(20) ADDRESS	4	ZZ1327WP	->BLSR327W (macro BLSR327W)
36	(24) ADDRESS	4	ZZ1ZZ2P	->ZZ2 (always addresses the master task variable)
40	(28) ADDRESS	4	ZZ1PDRP	->DMCB for problem directory
44	(2C) ADDRESS	4	ZZ1DSDP	->DMCB for data set directory
48	(30) ADDRESS	4	ZZ1DMICBP	->DMCB chain
52	(34) UNSIGNED	8		
60	(3C) UNSIGNED	4	ZZ1TLN	Terminal linesize
64	(40) UNSIGNED	4	ZZ1TPAGE	Terminal lines per screen
68	(44) CHARACTER	8	ZZ1PFILE	Default print file
76	(4C) UNSIGNED	4	ZZ1PGSZ	Default print page size
80	(50) UNSIGNED	4	ZZ1PLN	Default print line size
84	(54) UNSIGNED	4	ZZ1VSUPI	IPCSDDIR update interval
88	(58) UNSIGNED	4	ZZ1VSUPL	IPCSDDIR logical update count

BLSUZZ1 - Storage Layout (continued)

Offsets	Type	Length	Name	Description
92	(5C) UNSIGNED	4	ZZ1VSUPP	IPCSDDIR physical update count
96	(60) UNSIGNED	40		
136	(88) ADDRESS	4		
140	(8C) ADDRESS	4	ZZ1FPP	->FPBLOK (macro BLSFP)
144	(90) ADDRESS	224		
368	(170) BITSTRING	8	ZZ1F	Flags
	1... ..		ZZ1FU	In use
	.111 1111			
	1... ..		ZZ1FACCM	Access method determined
	.1... ..		ZZ1FTCAM	TCAM/TIOC access method
	..11 1111			
	1... ..		ZZ1FPRNT	Print file usable
	.111 1111			
371	(173) BITSTRING	5		
376	(178) BITSTRING	1	ZZ1USTAE	BLSUSTAE recursion control data
377	(179) BITSTRING	135		
512	(200) CHARACTER	0	ZZ199999	BLSUZZ1 #MD82236

BLSUZZ2

Common Name: Active Task Variable

Macro ID: BLSUZZ2

Created by:

- Compilation of module BLSUPUT
- Dynamically by module BLSUZZ2C and BLS9CMD1

Pointed to by: Field ZZ1ZZ2P (see “BLSUZZ1” on page 404) and field ZZ2ZZ2P (see “BLSUZZ2”)

Function: Contain information related to one task in the IPCS environment.

Note: The macro used to describe this structure permits the specification of the initial characters of each symbol. In this example, the characters ZZ2 were chosen.

Storage Layout

Offsets	Type	Length	Name	Description
0	(0) STRUCTURE	2048	ZZ2	BLSUZZ2 #MD83169
0	(0) CHARACTER	138	ZZ2AMD	
0	(0) ADDRESS	4	ZZ2AMDT	->Current TCB
4	(4) UNSIGNED	2	ZZ2AMDA	Address Space Identifier (ASID)
6	(6) UNSIGNED	1	ZZ2AMD0	Subpool identifier
7	(7) BITSTRING	1	ZZ2AMD1	Flags
1... ..			ZZ2AMD2	0=>SYSABEND, 1=>AMDPRDMP (or IPCS)
.1... ..				0=>VS2, 1=>VS1
..1.			ZZ2AMDI	0=>AMDPRDMP, 1=>IPCS
...1				
.... 1111				
8	(8) ADDRESS	4	ZZ2AMDB	->120-byte output buffer
12	(C) ADDRESS	4	ZZ2AMDP	->Print subroutine
16	(10) ADDRESS	4	ZZ2AMDC	->CVT
20	(14) ADDRESS	4	ZZ2AMDS	->Storage access subroutine
24	(18) ADDRESS	4	ZZ2AMDF	->Format subroutine
28	(1C) BITSTRING	16	ZZ2AMDU	Exit data area
44	(2C) ADDRESS	4	ZZ2AMD L	->Format pattern label
48	(30) ADDRESS	4	ZZ2AMDD	->Format pattern data
52	(34) ADDRESS	4	ZZ2AMDXP	->Parameter list extension
56	(38) BITSTRING	82		
138	(8A) CHARACTER	125	ZZ2PRT	Variable-length record
138	(8A) UNSIGNED	2	ZZ2PRTL	Record length
140	(8C) UNSIGNED	2	ZZ2PRT0	Offset
142	(8E) CHARACTER	121	ZZ2PRTT	ZZ2PRTT contains 2 sections
142	(8E) CHARACTER	1	ZZ2PRTT\$	ASA control character
143	(8F) CHARACTER	120	ZZ2PRTTA	Section 1
263	(107) CHARACTER	1		Buffer overflow area

BLSUZZ2 - Storage Layout (continued)

Offsets	Type	Length	Name	Description
264	(108) BITSTRING	8		
SETDEF data pertaining to this task				
272	(110) CHARACTER	656	ZZ2A	BLSUZZ3 #MD82026
272	(110) UNSIGNED	2	ZZ2ACH	Scan depth limit 0=>FLAG(I), 4=>FLAG(W), 8=>FLAG(E), 12=>FLAG(S), 16=>FLAG(T)
274	(112) UNSIGNED	1	ZZ2ASE	
275	(113) BITSTRING	1		
276	(114) UNSIGNED	4	ZZ2ALINE	Maximum output line size
280	(118) BITSTRING	8		
288	(120) BITSTRING	8	ZZ2AF	Binary options
Message switching options				
1...			ZZ2AFP	PRINT
.1...			ZZ2AFT	TERMINAL
..1.			ZZ2AFS	Full screen
...1	1111			
Message suppression options				
1...			ZZ2AFC	CONFIRM
.111				
....	1...		ZZ2AFV	VERIFY
....	.111			
Display content control options				
11..				
..1.			ZZ2AFM	DISPLAY(MACHINE)
...1				
....	1...		ZZ2AF\$	DISPLAY(REMARK)
....	.1..		ZZ2AFQ	DISPLAY(REQUEST)
....	..1.		ZZ2AFD	DISPLAY(STORAGE)
....	...1		ZZ2AFS	DISPLAY(SYMBOL)
291	(123) BITSTRING	2		
Miscellaneous options				
1...			ZZ2AFX	TEST
.111	1111			
1111	1111			
1111	1111			
296	(128) BITSTRING	40		

Default problem identifier

336	(150)	CHARACTER	8	ZZ2APID	ZZ2APID contains 2 sections
336	(150)	CHARACTER	3	ZZ2APIDA	Section 1
339	(153)	CHARACTER	5	ZZ2APIDB	Section 2
344	(158)	BITSTRING	2		

Default data set information

346	(15A)	CHARACTER	66	ZZ2AD	Default data set information
-----	-------	-----------	----	-------	------------------------------

Data set name

346	(15A)	CHARACTER	46	ZZ2ADD	BLRPRM #MD82025
346	(15A)	UNSIGNED	2	ZZ2ADDL	Length of ZZ2ADDT
348	(15C)	CHARACTER	44	ZZ2ADDT	BLRTEXT #MD82080
348	(15C)	CHARACTER	44	ZZ2ADDTA	Section 1

Member name

392	(188)	CHARACTER	10	ZZ2ADM	BLRPRM #MD82025
392	(188)	UNSIGNED	2	ZZ2ADML	Length of ZZ2ADMT
394	(18A)	CHARACTER	8	ZZ2ADMT	BLRTEXT #MD82080
394	(18A)	CHARACTER	8	ZZ2ADMTA	Section 1

Password

402	(192)	CHARACTER	10	ZZ2ADP	BLRPRM #MD82025
402	(192)	UNSIGNED	2	ZZ2ADPL	Length of ZZ2ADPT
404	(194)	CHARACTER	8	ZZ2ADPT	BLRTEXT #MD82080
404	(194)	CHARACTER	8	ZZ2ADPTA	Section 1

->ZZ6 (macro BLSRZZ6)

412	(19C)	ADDRESS	4	ZZ2AZZ6P	->ZZ6 (macro BLSRZZ6)
-----	-------	---------	---	----------	-----------------------

Default address space

416	(1A0)	CHARACTER	16	ZZ2AQAS	BLSRDATS #MD83116
416	(1A0)	CHARACTER	0	ZZ2AQAS0	BLSRDATS #MD83116

BLSUZZ2 - Storage Layout (continued)

Offsets	Type	Length	Name	Description
416	(1A0) CHARACTER	2	ZZ2AQAST	Address space type code
418	(1A2) BITSTRING	2		
420	(1A4) UNSIGNED	4	ZZ2AQAS1	Integer 1
424	(1A8) UNSIGNED	4	ZZ2AQAS2	Integer 2
428	(1AC) BITSTRING	4		
432	(1B0) CHARACTER	0	ZZ2AQAS9	BLSRDATS #MD83116

Default data characteristics

432	(1B0) CHARACTER	60	ZZ2AQD	BLSRDATC MF(SUBLIST,ZZ2AQD,,3)
432	(1B0) CHARACTER	0	ZZ2AQD00	BLSRDATC #MD83116
432	(1B0) SIGNED	4	ZZ2AQDOF	Offset in bytes
436	(1B4) UNSIGNED	4	ZZ2AQDLE	Length in bytes
440	(1B8) UNSIGNED	1	ZZ2AQDOB	BLSRDATT MF(SUBLIST,ZZ2AQDT,,4) BLSRDATT #MD83116 Data type code
441	(1B9) UNSIGNED	1	ZZ2AQDLB	
442	(1BA) CHARACTER	34	ZZ2AQDT	
442	(1BA) CHARACTER	0	ZZ2AQDT0	
442	(1BA) CHARACTER	1	ZZ2AQDTY	
443	(1BB) BITSTRING	1		
444	(1BC) CHARACTER	31	ZZ2AQDTD	Data name
475	(1DB) CHARACTER	1	ZZ2AQDTE	reserved
476	(1DC) CHARACTER	0	ZZ2AQDT9	BLSRDATT #MD83116
476	(1DC) UNSIGNED	4	ZZ2AQDIM	Array entry count
480	(1E0) SIGNED	4	ZZ2AQDIL	Subscript of initial array entry
484	(1E4) BITSTRING	4	ZZ2AQDF ZZ2AQDFA	Flags Array
	1... .. .iii iiii 1111 1111 1111 1111 1111 1111			
488	(1E8) BITSTRING	4		
492	(1EC) CHARACTER	0	ZZ2AQD99	BLSRDATC #MD83116
492	(1EC) BITSTRING	4		
496	(1F0) BITSTRING	432		
928	(3A0) CHARACTER	0	ZZ2A9999	BLSUZZ3 #MD82026

IPCS task identification data

928	(3A0) CHARACTER	8	ZZ2IDENT	Task name
936	(3A8) CHARACTER	8	ZZ2SC1DN	Subcommand name

BLSUZZ2 - Storage Layout (continued)

Offsets	Type	Length	Name	Description
944 (3B0)	CHARACTER	8	ZZ2SCMDE	Entry point name
952 (3B8)	BITSTRING	8		

Task AUTOMATIC storage control data (see module BLSUALL0)

960 (3C0)	BITSTRING	8	ZZ2STG	AUTOMATIC storage control data
960 (3C0)	ADDRESS	4	ZZ2STGP	->Current extent
964 (3C4)	UNSIGNED	4	ZZ2STGN	Number of bytes available

Task TOD clock value

968 (3C8)	BITSTRING	8	ZZ2TOD	Task TOD clock value
976 (3D0)	BITSTRING	16		

Subroutine addresses

992 (3E0)	ADDRESS	4	ZZ2ALLOP	->BLSUALL0
996 (3E4)	ADDRESS	4		reserved
1000 (3E8)	ADDRESS	4	ZZ2FF19P	->IKJEFF19
1004 (3EC)	ADDRESS	4	ZZ2FRE1P	->BLSUFRE1
1008 (3F0)	ADDRESS	4	ZZ2FF18P	->IKJEFF18
1012 (3F4)	ADDRESS	4	ZZ2DYNAP	->BLSUDYNA
1016 (3F8)	ADDRESS	4	ZZ2ZZ2CP	->BLSUZZ2C
1020 (3FC)	ADDRESS	4	ZZ2ZZ2DP	->BLSUZZ2D
1024 (400)	ADDRESS	4	ZZ2STAIP	->BLSUSTAI
1028 (404)	ADDRESS	4	ZZ2FF02P	->IKJEFF02
1032 (408)	ADDRESS	4	ZZ2GETLP	->IKJGETL
1036 (40C)	ADDRESS	4	ZZ2PARSP	->IKJPARS
1040 (410)	ADDRESS	4	ZZ2PTGTP	->IKJPTGT
1044 (414)	ADDRESS	4	ZZ2PUTLP	->IKJPUTL
1048 (418)	ADDRESS	4	ZZ2SCANP	->IKJSCAN
1052 (41C)	ADDRESS	4	ZZ2STCKP	->IKJSTCK
1056 (420)	ADDRESS	4	ZZ2AL0CP	->BLSAL0C
1060 (424)	ADDRESS	4	ZZ2MSGSP	->BLSDM1SGS

BLSUZZ2 - Storage Layout (continued)

Offsets	Type	Length	Name	Description
1064 (428)	ADDRESS	4		
1068 (42C)	ADDRESS	4	ZZ2TRMVP	->BLSUTRMV
1072 (430)	ADDRESS	4	ZZ2FT25P	->IKJEFT25
1076 (434)	ADDRESS	4	ZZ2MSG0P	->BLSDMSG0
1080 (438)	ADDRESS	4	ZZ2PROPP	->BLSUPROP
1084 (43C)	ADDRESS	4	ZZ2PUTNP	->BLSUPUTN
1088 (440)	ADDRESS	4	ZZ2TRMNP	->BLSUTRMN
1092 (444)	ADDRESS	4	ZZ2ENQ0P	->BLSDENQ0
1096 (448)	ADDRESS	4	ZZ2ADSDP	->BLSDADSD
1100 (44C)	ADDRESS	4	ZZ2APDRP	->BLSDAPDR
1104 (450)	ADDRESS	4		
1108 (454)	ADDRESS	4	ZZ2FPDRP	->BLSDFPDR
1112 (458)	ADDRESS	4	ZZ2MPKNP	->BLSUMPKN
1116 (45C)	ADDRESS	4	ZZ2MPK1P	->BLSUMPK1
1120 (460)	ADDRESS	4		
1124 (464)	ADDRESS	4		
1128 (468)	ADDRESS	4	ZZ2PRTAP	->BLSUPRTA
1132 (46C)	ADDRESS	4	ZZ2PRTNP	->BLSUPRTN
1136 (470)	ADDRESS	4	ZZ2PRTTP	->BLSUPRTT
1140 (474)	ADDRESS	16		
1156 (484)	ADDRESS	4	ZZ2PUTTP	->BLSUPUTT
1160 (488)	ADDRESS	4	ZZ2PUTIP	->BLSUPUTI
1164 (48C)	ADDRESS	4	ZZ2TRMAP	->BLSUTRMA
1168 (490)	ADDRESS	4	ZZ2VSCAP	->BLSUVSCA
1172 (494)	ADDRESS	4	ZZ2ACCIP	->BLSRACCI
1176 (498)	ADDRESS	4	ZZ2MOATP	->BLSUMOAT
1180 (49C)	ADDRESS	4	ZZ2STKSP	->BLSUSTKS
1184 (4A0)	ADDRESS	4	ZZ2LINKP	->ISPLINK
1188 (4A4)	ADDRESS	4	ZZ2STDEP	->BLSUSTDE
1192 (4A8)	ADDRESS	4	ZZ2VARXP	->BLSUVARX
1196 (4AC)	ADDRESS	52		

Task data addresses

BLSUZZ2 - Storage Layout (continued)

Offsets	Type	Length	Name	Description
1248 (4E0)	ADDRESS	4	ZZ2ZZ2P	->Next task variable
1252 (4E4)	ADDRESS	4	ZZ2STBLP	->Subcommand table
1256 (4E8)	ADDRESS	4	ZZ2TCBP	->OS TCB for task
1260 (4EC)	ADDRESS	4	ZZ2RPLP	->Active RPL
1264 (4F0)	ADDRESS	4	ZZ2ZZ1P	->ZZ1 (macro BLSUZZ1)
1268 (4F4)	ADDRESS	4	ZZ2ZZ6P	->ZZ6 chain (macro BLSRZZ6)
1272 (4F8)	ADDRESS	4	ZZ2RZZ6P	->ZZ6 on entry to ABEND
1276 (4FC)	ADDRESS	4	ZZ2DVTP	->DVT service vector
1280 (500)	ADDRESS	4	ZZ2BVTP	->BVT service vector
1284 (504)	ADDRESS	4	ZZ2RVTP	->RVT service vector
1288 (508)	ADDRESS	4	ZZ2VRSAP	->Active record block for SA record
1292 (50C)	CHARACTER	16	ZZ2CPPL	CPPL (macro IKJCPPL)
1292 (50C)	ADDRESS	4	ZZ2CPPLC	->Subcommand buffer
1296 (510)	ADDRESS	4	ZZ2CPPLU	->UPT (macro IKJUPT)
1300 (514)	ADDRESS	4	ZZ2CPPLP	->PSCB (macro IKJPSCB)
1304 (518)	ADDRESS	4	ZZ2CPPLE	->ECT (macro IKJECT)
1308 (51C)	ADDRESS	4	ZZ2VRESP	->Active record block for ES record
1312 (520)	ADDRESS	4	ZZ2LOADP	->EPT (macro BLSUPGMT)
1316 (524)	ADDRESS	4	ZZ2TTLP	->Print file subtitle chain
1320 (528)	ADDRESS	4	ZZ2SCMP	->Subcommand work area
1324 (52C)	ADDRESS	4	ZZ200SYP	->LOCO (macro BLSU00SY)
1328 (530)	ADDRESS	4	ZZ2ESSYP	->ESCO (macro BLSRESSY)
1332 (534)	ADDRESS	4	ZZ2PHSYP	->PHCO (macro BLSRPHSY)
1336 (538)	ADDRESS	4	ZZ2RASYP	->RACO (macro BLSRRASY)
1340 (53C)	ADDRESS	4	ZZ2RDSYP	->RDCO (macro BLSRRDSY)
1344 (540)	ADDRESS	4	ZZ2SASYP	->SACO (macro BLSRSASY)
1348 (544)	ADDRESS	4	ZZ2FFSYP	->FFCO (macro BLSUFFSY)
1352 (548)	ADDRESS	4	ZZ2SDSYP	->SDCO (macro BLSUSDSY)
1356 (54C)	ADDRESS	4	ZZ2VRSYP	->Active record block for BLSRESGE
1360 (550)	ADDRESS	4	ZZ2COPYP	->COPYDUMP data block
1364 (554)	CHARACTER	16	ZZ2IOPL	IOPL (macro IKJIOPL)
1364 (554)	ADDRESS	4	ZZ2IOPLU	->UPT (macro IKJUPT)
1368 (558)	ADDRESS	4	ZZ2IOPLE	->ECT (macro IKJECT)

BLSUZZ2 - Storage Layout (continued)

<u>Offsets</u>	<u>Type</u>	<u>Length</u>	<u>Name</u>	<u>Description</u>
1372 (55C)	ADDRESS	4	ZZ2IOPLC	->Event Control Block
1376 (560)	ADDRESS	4	ZZ2IOPLI	->Service routine block
1380 (564)	ADDRESS	4	ZZ2TVTP	->TVT service vector
1384 (568)	ADDRESS	4	ZZ2RBSYP	->RBCO (macro BLSURBSY)
1388 (56C)	ADDRESS	4	ZZ2RCSYP	->RCCO (macro BLSURCSY)
1392 (570)	ADDRESS	4	ZZ2BLDLP	->PGB (macro BLSUBLDT)
1396 (574)	ADDRESS	4	ZZ2RV2P	->RV2 service vector
1400 (578)	ADDRESS	4	ZZ2ZZ0P	->ZZ0 (macro BLSUZZ0)
1404 (57C)	ADDRESS	4	ZZ2ZZ0TP	->ZZ0T (macro BLSUZZ0T)
1408 (580)	ADDRESS	4	ZZ2NTRCP	->NTRC (macro BLSLNTRC)
1412 (584)	ADDRESS	4	ZZ2QSLLP	->Services load list
1416 (588)	ADDRESS	80		
1496 (5D8)	ADDRESS	4	ZZ2PRXP	->PRX (macro BLSRPRX)
1500 (5DC)	ADDRESS	4		reserved

Numeric variables for the task

1504 (5E0)	UNSIGNED	20		
1524 (5F4)	UNSIGNED	2	ZZ2CW	Current scan depth
1526 (5F6)	UNSIGNED	2	ZZ2NEST	Application Nesting Level
1528 (5F8)	BITSTRING	8	ZZ2CUST1	Customer modification data

Miscellany

1536 (600)	CHARACTER	8	ZZ2PGM	BLSULINK program name
1544 (608)	CHARACTER	8	ZZ2CCHAR	BLSUCNV CHAR data
1544 (608)	BITSTRING	8	ZZ2CPACK	BLSUCNV PACKED data
1552 (610)	CHARACTER	16	ZZ2CZONE	BLSUCNV ZONED data
1568 (620)	ADDRESS	4	ZZ2CPTR	BLSUCNV PTR data
1568 (620)	BITSTRING	4	ZZ2CPTRB	Bit string overlay
1572 (624)	BITSTRING	4	ZZ2SPIET	ESPIE token
1576 (628)	BITSTRING	416		
1992 (7C8)	BITSTRING	8	ZZ2ATTN	Attention processing data
1992 (7C8)	ADDRESS	4	ZZ2ACMP	Attention exit subcommand address



BLSUZZ2 - Storage Layout (continued)

Offsets	Type	Length	Name	Description
1....			ZZ2ITSEW	WAIT
.1... ..			ZZ2ITSEP	POST
..11 1111				
2041	(7F9) UNSIGNED	3	ZZ2ITSEC	Completion code
2044	(7FC) BITSTRING	4		
2048	(800) CHARACTER	0	ZZ299999	BLSUZZ2 #MD83169

Alphabetic Index to Field Names

Name	Offset	Name	Offset	Name	Offset
ZZ2	0	ZZ2A	110	ZZ2ACCIP	494
ZZ2ACH	110	ZZ2ACMP	7C8	ZZ2AD	15A
ZZ2ADD	15A	ZZ2ADDL	15A	ZZ2ADDT	15C
ZZ2ADDTA	15C	ZZ2ADMI	188	ZZ2ADML	188
ZZ2ADMT	18A	ZZ2ADMTA	18A	ZZ2ADP	192
ZZ2ADPL	192	ZZ2ADPT	194	ZZ2ADPTA	194
ZZ2ADSDP	448	ZZ2AF	120	ZZ2AF\$	122 08
ZZ2AFC	121 80	ZZ2AFD	122 02	ZZ2AFFS	120 20
ZZ2AFM	122 20	ZZ2AFP	120 80	ZZ2AFQ	122 04
ZZ2AFS	122 01	ZZ2AFT	120 40	ZZ2AFV	121 08
ZZ2AFX	125 80	ZZ2ALINE	114	ZZ2ALLOP	3E0
ZZ2ALOCF	420	ZZ2AMD	0	ZZ2AMDA	4
ZZ2AMDB	8	ZZ2AMDC	10	ZZ2AMDD	30
ZZ2AMDF	18	ZZ2AMDI	7 10	ZZ2AMDLE	2C
ZZ2AMDP	C	ZZ2AMDS	14	ZZ2AMDT	0
ZZ2AMDU	1C	ZZ2AMDXP	34	ZZ2AMDO	6
ZZ2AMD1	7 80	ZZ2AMD2	7 40	ZZ2APDRP	44C
ZZ2APID	150	ZZ2APIDA	150	ZZ2APIDB	153
ZZ2AQAS	1A0	ZZ2AQASt	1A0	ZZ2AQAS0	1A0
ZZ2AQAS1	1A4	ZZ2AQAS2	1A8	ZZ2AQAS9	1B0
ZZ2AQD	1B0	ZZ2AQDF	1E4	ZZ2AQDFA	1E4 80
ZZ2AQDIL	1E0	ZZ2AQDIM	1DC	ZZ2AQDLB	1B9
ZZ2AQDLE	1B4	ZZ2AQDOB	1B8	ZZ2AQDOF	1B0
ZZ2AQDT	1BA	ZZ2AQDTE	1BC	ZZ2AQDTE	1DB
ZZ2AQDTY	1BA	ZZ2AQDT9	1BA	ZZ2AQDT9	1DC
ZZ2AQD00	1B0	ZZ2AQD99	1EC	ZZ2ASE	112
ZZ2ATTN	7C8	ZZ2AZZ6P	19C	ZZ2A9999	3A0
ZZ2BLDLP	570	ZZ2BVTP	500	ZZ2CCHAR	608
ZZ2COPYP	550	ZZ2CPACK	608	ZZ2CPPL	50C
ZZ2CPPLC	50C	ZZ2CPPLE	518	ZZ2CPPLP	514
ZZ2CPPLU	510	ZZ2CPTB	620	ZZ2CPTRB	620
ZZ2CUST1	5F8	ZZ2CW	5F4	ZZ2CZONE	610
ZZ2DVTP	4FC	ZZ2DYNAP	3F4	ZZ2ENQOP	444
ZZ2ESSYP	530	ZZ2EVE	7CC	ZZ2EVEC	7CD
ZZ2EVEP	7CC 40	ZZ2EVEW	7CC 80	ZZ2F	7D0
ZZ2FFSYP	544	ZZ2FF02P	404	ZZ2FF18P	3F0
ZZ2FF19P	3E8	ZZ2FM	7D0 20	ZZ2FNENV	7D0 10
ZZ2FPDRP	454	ZZ2FRE1P	3EC	ZZ2FSPIE	7D0 08
ZZ2FSPIN	7D0 04	ZZ2FT25P	430	ZZ2FVARX	7D0 02
ZZ2FX	7D0 40	ZZ2F3270	7D0 80	ZZ2GETLP	408
ZZ2IDENT	3A0	ZZ2IOPL	554	ZZ2IOPLC	55C
ZZ2IOPLE	558	ZZ2IOPLI	560	ZZ2IOPLU	554
ZZ2ITR	7E0	ZZ2ITRE	7F0	ZZ2ITREC	7F1
ZZ2ITREP	7F0 40	ZZ2ITREW	7F0 80	ZZ2ITRPP	7E0
ZZ2ITRRC	7E8	ZZ2ITR2P	7E4	ZZ2ITSE	7F8
ZZ2ITSEC	7F9	ZZ2ITSEP	7F8 40	ZZ2ITSEW	7F8 80
ZZ2LINKP	4A0	ZZ2LOADP	520	ZZ2IIOATP	498
ZZ2MPKNP	458	ZZ2MPK1P	45C	ZZ2MSGSP	424
ZZ2MSGOP	434	ZZ2NEST	5F6	ZZ2NTRCP	580
ZZ2PARSP	40C	ZZ2PGM	600	ZZ2PHSYP	534
ZZ2PROPP	438	ZZ2PRT	8A	ZZ2PRTAP	468
ZZ2PRTL	8A	ZZ2PRTNP	46C	ZZ2PRT0	8C

BLSUZZ2 - Alphabetic Index to Field Names (continued)

<u>Name</u>	<u>Offset</u>	<u>Name</u>	<u>Offset</u>	<u>Name</u>	<u>Offset</u>
ZZ2PRTT	8E	ZZ2PRTT\$	8E	ZZ2PRTTA	8F
ZZ2PRTTP	470	ZZ2PRXP	5D8	ZZ2PTGTP	410
ZZ2PUTIP	488	ZZ2PUTLP	414	ZZ2PUTNP	43C
ZZ2PUTTP	484	ZZ2QSLLP	584	ZZ2RASYP	538
ZZ2RBSYP	568	ZZ2RC5YP	56C	ZZ2RDSYP	53C
ZZ2RPLP	4EC	ZZ2RSTAE	7DF	ZZ2RVTP	504
ZZ2RV2P	574	ZZ2RZZ6P	4F8	ZZ2SASYP	540
ZZ2SCANP	418	ZZ2SCMDE	3B0	ZZ2SCMDN	3A8
ZZ2SCWP	528	ZZ2SDSYP	548	ZZ2SPIET	624
ZZ2STAIP	400	ZZ2STBLP	4E4	ZZ2STCKP	41C
ZZ2STDEP	4A4	ZZ2STG	3C0	ZZ2STGN	3C4
ZZ2STGP	3C0	ZZ2STKSP	49C	ZZ2TCBP	4E8
ZZ2TOD	3C8	ZZ2TRMAP	48C	ZZ2TRMNP	440
ZZ2TRMVP	42C	ZZ2TTLP	524	ZZ2TVTP	564
ZZ2VARXP	4A8	ZZ2VRESP	51C	ZZ2VRSAP	508
ZZ2VRSYP	54C	ZZ2VSCAP	490	ZZ2ZZ0P	578
ZZ2ZZ0TP	57C	ZZ2ZZ1P	4F0	ZZ2ZZ2CP	3F8
ZZ2ZZ2DP	3FC	ZZ2ZZ2P	4E0	ZZ2ZZ6P	4F4
ZZ2005YP	52C	ZZ299999	800		

Section 5. Diagnostic Aids

This section contains information that can help in diagnosing problems within IPCS. It includes:

- A return code table that shows which return codes are issued by each IPCS module. The modules are listed in alphabetic order.
- The user completion (ABEND) codes and problem determination TABLE I.
- Two ABEND code tables, the first sorted by code, the second sorted by module.
- A discussion of error recovery.

IPCS Return Code Table

IPCS modules generally produce the following return codes:

Code	Meaning of Code
0	The request was processed normally.
4	One or more warning conditions (unusual but not necessarily erroneous) were detected, but they did not prevent the request from being processed.
8	One or more error conditions were detected, but they did not prevent the request from being processed.
12	Processing was terminated because of an event that was anticipated. The most common event taht causes this return code is the use of the attention interrupt to terminate a subcommand or CLIST.
16	Processing was terminated because of an environmental error such as an I/O error or a lack of virtual storage needed to process the request.
***	See the individual subcommand in the <i>IPCS User's Guide and Reference</i> book for an explanation of these return codes.

Figure 128. Standard Return Code Summary

Those IPCS modules that produce the preceding return codes for other reasons and those modules that produce other return codes are included alphabetically in the following list along with a summary of their return codes:

Module	Code	Meaning of the code
BLSCAAMS	0	Successful
	4	Warning condition encountered
	8	Termination because of an error condition
	12	Termination because of insufficient space
BLSCABLD	0	Successful
	8	Termination because of an error condition
BLSCADST	0	Successful
	16	Termination because of an invalid request
BLSCADYN	0	Successful
	8	Termination because of an error condition
	12	Termination because of insufficient space
BLSCAINT	0	Successful
	8	Termination because of an unavailable model
BLSCALOC	0	Successful
	4	Warning condition encountered
	8	Termination because of an error condition
	12	Termination because of insufficient space
	16	Termination because of an invalid DCMB
BLSCAMRE	0	Successful
	8	Termination because of an error condition
BLSCAMIN	0	Successful
	4	End of data for GET operation
	8	Error condition encountered
	12	No further calls allowed
BLSCAMPR	0	Successful
	12	No further calls allowed
BLSCANAL	0	Successful
	8	Error from DAIRFAIL module
	12	Termination because of insufficient space
BLSCANLE	0	Successful
	8	Error from VSAMFAIL module
	12	Termination because of insufficient space
BLSCANLI	0	Successful
	8	Link to IKJEFF02 failed
	12	Termination because of insufficient space
	76	Parameter list error
BLSCCLSE	0	Successful
	4	Permanent I/O error encountered
BLSCENDD	0	Successful
	4	DMCB not opened
BLSCRESE	0	Successful
	4	Invalid request
	8	Record not held for update
	12	Permanent error

BLSCFAMS	0	Successful
	4	Warning condition encountered
	8	Termination because of an error
BLSCFDYN	0	Successful
	4	Warning condition encountered
	8	Termination because of an error
	12	Termination because of insufficient space
BLSCFREE	0	Successful, data set freed
	4	Execution may be incomplete
	8	Termination because of an error
BLSCGETT	0	Successful
	4	Invalid request
	8	End of file or record not found
	12	Permanent error
BLSCOPEN	0	Successful
	4	Termination because of an attention interrupt
	8	Failure, but DMCB restored
	12	Failure and DMCB not restored
BLSCPOIN	0	Successful
	4	Invalid request
	8	Record not found
	12	Permanent error
BLSCPUTT	0	Successful
	4	Invalid request
	8	Duplicate record key found on addition
	12	Permanent error
BLSCRFM	0	Successful
	12	Message BLS03109 issued
BLSCRQST	0	Successful
	4	One of the following: 1. Attention interrupt 2. Permanent error on CLOSE
	8	One of the following: 1. Recoverable error on OPEN 2. End-of-file or record not found on GET or POINT 3. Duplicate key on PUT 4. Record not held for update on REASE
	12	Permanent error
BLSCSETT	0	Successful
	4	Invalid request
BLSCVALT	0	Successful
	8	Enqueue conflict
BLSDADSD	0	Successful
	8	Allocation failed
	16	Facility parameter block not available
BLSDAPDR	0	Successful
	8	Allocation failed
	16	Facility parameter block not available
BLSDC070	0	Successful
	8	Parameter invalid
BLSDC128	0	Successful
	8	Parameter invalid

Module	Code	Meaning of the code
BLSDC600	0	Successful
	8	Parameter invalid
BLSDDT00	0	Successful
	4	Parameter invalid
BLSDFDSD	0	Successful
	4	Module task fails to match OPEN TCB
	8	CLOSE or FREE failed
BLSDFPDR	0	Successful
	4	Module task fails to match OPEN TCB
	8	CLOSE or FREE failed
BLSDPC00	0	Successful
	4	Parameter invalid
BLSDSV00	0	Successful
	4	Parameter invalid
BLSDTM00	0	Successful
	4	Parameter invalid
BLSDWAIT	0	Successful
	4	Attention interrupt
BLSEAUTH	0	Successful, user authorized
	4	User not authorized
BLSEDEL0	***	DELPROB subcommand return codes
BLSELPCC	0	Successful
	4	Record does not meet criteria
BLSELPCL	0	Successful
	8	Error condition in called routine
	12	Severe error in called routine
BLSELPCT	0	Successful
	12	Return code of 4 from BLSOPEN
BLSFAV00	0	Successful
	8	Member present and MANAGED requested
	12	Invalid or missing problem number or data set name
BLSFCONF	0	Successful
	8	Completed, missing DSD base record encountered
	12	Termination because of an attention interrupt
BLSFDALL	0	All associations deleted and all records found
	4	All associations deleted, but some records missing
	8	Attention interrupt, each association deleted or left intact
BLSFDDDP	0	Successful, all records found
	4	Successful, but missing records in the data base
	8	Attention interrupt, data base left in consistent state
BLSFDEL	0	Successful deletion, all records found
	4	Delete performed, but records missing
	8	Attention interrupt, no records deleted
BLSFFL00	0	Successful
	12	Record is not a base record

Module	Code	Meaning of the code
BLSFFP00	0	Successful, both records available
	4	No association, neither record available
	8	Problem association record only, available
	12	Termination because of an attention interrupt
BLSFLALL	0	Successful
	4	Successful, empty list built and no associations found
	8	Attention interrupt, no list built
BLSFOD00	0	No conflicts exist
	4	Conflicts overridden
	12	Conflicts not overridden
BLSFOPEN	0	Successful
	4	Termination because of an attention interrupt
BLSFPARS	0	Successful
	8	No problem ID or DSN specified and no default
	12	Severe error encountered, no PDL built
BLSFPREA	0	Successful
	4	Sequence does not exist
BLSFPRES	0	Successful
	4	Record does not exist
BLSFSCRT	0	Data set scratched
	4	Data set initialized
	12	Failure to scratch data set
BLSFSD00	0	Successful
	8	Unable to set default
BLSFTL00	0	Successful
	12	Record not a base record or access error error occurred
	16	Termination because of called routines
BLSFVRFY	0	Authorization has been checked
	4	Authorization has not been checked because of a missing record
	12	Termination because of an attention interrupt
BLSLIADS	0	Normal completion
	12	Invalid input
	16	Termination
BLSLIDSN	0	Normal completion
	12	Invalid input
	16	Termination
BLSQCFMT	0	Normal completion
	4	Warning
	16	Termination - unable to obtain storage for the CBAT
BLSQEXTI	0	Normal completion
	16	Termination because of a failure to load a service or a failure to obtain storage for load list
BLSQFORM	0	Normal completion
	4	Truncation of data before end of model
	8	Error in control block model detected

BLSQIFMT	0	Successful
	4	Warning
	16	Termination because of an invalid model specification
BLSQSEL	0	Normal completion
	4	Jobname not found or the requested ASID was not assigned
	8	Dump access failures were encountered
	16	Termination because of one of the following:
		.CVT pointer is zero .Dump access failure for the CVT or the ASVT .Not enough virtual storage obtained Therefore, no output is produced
BLSQSIGR	0	Normal completion
	4	Warning - only the origin range was specified; end range is set to the origin
	12	Serious - specification error prevented processing
	16	Termination - IPCS environmental error
BLSRACCI	0	Successful
	4	Storage not available
BLSRACCL	0	Successful
	4	Storage not available
BLSRASCX	***	ASCBEXIT subcommand return codes
BLSRCOMP	***	COMPARE subcommand return codes
BLSRCOPY	***	COPYDUMP subcommand return codes
BLSREVAL	***	EVALUATE subcommand return codes
BLSRIOSK	***	IOSCHECK subcommand return codes
BLSRSETD	***	SETDEF subcommand return codes
BLSRTCXB	***	TCBEXIT subcommand return codes
BLSRVPAD	0	Address unqualified
	4	Address qualified
BLSRVPAS	0	Successful
	4	Invalid ASID
BLSRVPCP	0	Successful
	4	Invalid CPU address
BLSRVPTC	0	Successful
	4	Invalid TCB name
BLSRVPVS	0	Member not specified
	4	Member specified
BLSRVRBX	***	VERBEXIT subcommand return codes
BLSU	any	IPCS session return code
BLSUINI2	any	IPCS session return code
BLSUMOAT	any	Return code from attached task
BLSUMON	any	Session return code
BLSUMONA	0	Return code from attached command

Module	Code	Meaning of the code
BLSUMONC	0	Return code from EXEC command
BLSUMOII	0	Successful
	4	Error condition encountered
	16	Termination
BLSUMONI	0	Successful
	4	Error condition encountered
	16	Termination
BLSUMONL	16	Subcommand return code
BLSUMONT	0	Return code from TSO command
BLSUMONX	16	Return code from command or EXEC
BLSUPUTL	0	Successful
	4	Message not transmitted
BLSUVPVA	0	Successful
	4	Invalid value
BLSUVPX1	0	Successful
	4	Invalid hexadecimal number
BLSUVP21	0	Successful
	4	Error condition encountered
BLSUVP31	0	Successful, valid integer
	4	Warning, integer failed validation
BLSUVP32	0	Successful, valid integer
	4	Warning, integer failed validation
BLSUVP99	0	Successful, valid integer
	4	Warning, integer failed validation
BLS9MOII	0	Successful
	4	Error condition encountered
	16	Termination
BLS9MONI	0	Successful
	4	Error condition encountered
	16	Termination

IPCS User Completion (ABEND) Codes

The IPCS user completion ABEND codes are presented in hexadecimal order with the decimal equivalent. Each code is explained, and, where appropriate, the accompanying actions by the IPCS component are described and a programmer response is suggested. See the *System Messages* publication for information about the IPCS messages.

Problem determination actions accompany problem-identifying codes. Problem Determination Table I follows the user completion ABEND codes.

064 (decimal 0100)

Explanation: An unexpected error occurred while attempting to open the problem or data set directory for reading.

System Action: The ABEND follows messages that indicate the nature of the error. The subcommand terminates.

Programmer Response: Follow the programmer response for the BLS03nnnt messages that precede the ABEND.

Problem Determination: Table I, items 4, 5, 8.

065 (decimal 0101)

Explanation: An unexpected error occurred while attempting to get a record from the problem or data set directory.

System Action: The ABEND is preceded by a set of messages that indicates in more detail the nature of the error. The subcommand is terminated.

Programmer Response: Follow the programmer response for the BLS03nnnt messages that precede the ABEND.

Problem Determination: Table I, items 4, 5, 8.

066 (decimal 0102)

Explanation: An unexpected error occurred while attempting to close the problem or data set directory.

System Action: The ABEND is preceded by a set of messages that indicates in more detail the nature of the error. The subcommand is terminated.

Programmer Response: Follow the programmer response for the BLS03nnnt messages that precede the ABEND.

Problem Determination: Table I, items 4, 5, 8.

067 (decimal 0103)

Explanation: Internal error; Data Access Services was unable to issue the GETMAIN failure message. The precise error is indicated via the reason code located in general purpose register 15. Reason codes are:

Reason	Explanation
20	No component table entry; the requested message does not exist.
24	Zero component table entry; the requested message does not exist.
28	No subcomponent table entry; the requested message does not exist.
32	Zero subcomponent table entry; the requested message does not exist.
36	No sub-subcomponent table entry; the requested message does not exist.
40	Zero sub-subcomponent table entry; the requested message does not exist.
60	The reserved storage area was not large enough to contain the output message.
64	The GETMAIN failure message build routine does not permit the message to contain inserts.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

068 (decimal 0104)

Explanation: Module BLSDENQP detected an attempt to violate the IPCS resource serialization protocol; specifically, an attempt was made to acquire a problem resource while either the problem directory or the data set directory was open.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

069 (decimal 0105)

Explanation: Module BLSDMSG0 was called to produce a message, but during the message production procedure an error was detected. The precise error found is indicated via the reason code located in general purpose register 15. Reason codes are:

Reason	Explanation
4-16	A return code from module BLSUPUTN, BLSUPRTN, or BLSUTRMN, depending upon the routing specified.
20	No component table entry; the requested message does not exist.
24	Zero component table entry; the requested message does not exist.
28	No subcomponent table entry; the requested message does not exist.
32	Zero subcomponent table entry; the requested message does not exist.
36	No sub-subcomponent table entry; the requested message does not exist.
40	Zero sub-subcomponent table entry; the requested message does not exist.
44	The current message segment caused the message to exceed the maximum allowed length.
48	The current message insert caused the message to exceed the maximum allowed length.
52	The message skeleton required more inserts than were supplied by the caller.
56	The caller supplied more inserts than were required by the message skeleton.
60	The GETMAIN macro returned a nonzero return code while obtaining storage for the current output message.

System Action: The message is not issued. The message chain is not FREEMAINED. The subcommand is terminated.

Programmer Response: None.

Problem Determination: If the reason code in general purpose register 15 is 60, increase the region size and try again. If the error persists, or if the reason code is other than 60, perform Table I, items 4, 5, 8.

06A (decimal 0106)

Explanation: An I/O error occurred in module BLSEDEL0 during a DELPROB subcommand. The type of error is indicated by one of the following messages: BLS04042I, BLS04043I, or BLS04045I and their associated messages.

System Action: The DELPROB subcommand is terminated.

Programmer Response: None.

Problem Determination: Retry the DELPROB subcommand. If the error persists, perform Table I, items 1, 2, 3, 5, 6a, 7c (for IPCSPRnn), 8, 9a.

06B (decimal 0107)

Explanation: Internal error; module BLSEAUTH was called with an invalid authorization rule. Valid rules are O for owner and D for delete.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

06C (decimal 0108)

Explanation: An error was detected by module BLSEDELC while performing an open or close operation on the problem directory during confirmation processing.

System Action: The ABEND is preceded either by message BLS04042I or BLS04045I. The DELPROB subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 1, 2, 3, 5, 6a, 7c (for IPCSPRnn), 8, 9a.

06D (decimal 0109)

Explanation: A dump-directory equate-symbol record was passed to a service routine to have an action performed on its contents. The service routine, prior to performing the requested action, called module BLSRESCK to determine whether the record contents were proper. Module BLSRESCK determined that the record contents were improper. The ABEND is issued only if the TEST operand is specified.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

06E (decimal 0110)

Explanation: An invalid parameter was passed to a Data Access Services module.

System Action: The ABEND is preceded by a message that indicates the type of internal error detected. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

06F (decimal 0111)

Explanation: The records passed to module BLSFOD00 for comparison do not contain the same VSAM key.

System Action: The ABEND is preceded by message BLS04064I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

070 (decimal 0112)

Explanation: A logic error was detected by module BLSFOD00. The record passed to module BLSFOD00 for comparison was not a base record.

System Action: The ABEND is preceded by message BLS04061I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

071 (decimal 0113)

Explanation: During the opening of the data set directory, the DMCB (data access services control block) was determined by module BLSFUD00 to be unrecoverable. An attempt to free the data set directory was unsuccessful.

System Action: The ABEND is preceded by message BLS04052I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 1, 2, 3, 5, 6a, 7c (for IPCSPRnn), 8, 9a.

072 (decimal 0114)

Explanation: The user instructed the IPCS attention processor to issue a diagnostic abend by typing in ABEND.

System Action: The IPCS session is terminated.

Programmer Response: The purpose of the IPCS attention-exit ABEND function is to permit diagnostic information to be collected regarding apparent IPCS problems. Use the dump to confirm this diagnosis.

073 (decimal 0115)

Explanation: An error was detected while reading the data set directory.

System Action: The ABEND is preceded by message BLS04053I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 1, 2, 3, 5, 6a, 7c (for IPCSPRnn), 8, 9a.

074 (decimal 0116)

Explanation: An invalid write request was detected by module BLSFUD00 while writing a record to the data set directory.

System Action: The ABEND is preceded by message BLS04054I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 1, 2, 3, 5, 6a, 7c (for IPCSPRnn), 8, 9a.

075 (decimal 0117)

Explanation: A permanent write error was detected by module BLSFUD00 while writing a record to the data set directory.

System Action: The ABEND is preceded by message BLS04054I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Obtain the dump and determine the cause of the permanent write error. If the error persists, perform Table I, items 1, 2, 3, 6a, 7c (for IPCSPRnn), 8, 9a.

076 (decimal 0118)

Explanation: Module BLSFUD00 detected that the DMCB (data access services control block) is invalid while attempting to write a record to the data set directory.

System Action: The ABEND is preceded by informational message BLS04054I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

077 (decimal 0119)

Explanation: A permanent error was detected by module BLSFUD00 while attempting to close the data set directory.

System Action: The ABEND is preceded by message BLS04055I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 1, 2, 3, 5, 6a, 7c (for IPCSPRnn), 8, 9a.

078 (decimal 0120)

Explanation: An invalid DMCB (data access services control block) was detected by module BLSFUD00 while attempting to close the data set directory.

System Action: The ABEND is preceded by informational message BLS04055I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

079 (decimal 0121)

Explanation: Module BLSFUP00 detected an invalid DMCB (data access services control block) while

attempting to open the problem directory. The attempt to free the problem directory was unsuccessful.

System Action: The ABEND is preceded by informational message BLS04042I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

07A (decimal 0122)

Explanation: A GETMAIN macro instruction returned with a nonzero return code. Enough virtual storage could not be obtained.

System Action: The ABEND is preceded by message BLS04015I. The subcommand is terminated.

Programmer Response: Ensure that the region size is large enough for IPCS execution.

Problem Determination: Table I, items 4, 5, 8.

07B (decimal 0123)

Explanation: A module detected an error while reading the problem directory.

System Action: The ABEND is preceded by message BLS04043I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 1, 2, 3, 5, 6a, 7c (for IPCSPRnn), 8, 9a.

07C (decimal 0124)

Explanation: Module BLSFUP00 detected an invalid write request while attempting to write to the problem directory.

System Action: The ABEND is preceded by message BLS04044I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 1, 2, 3, 5, 6a, 7c (for IPCSPRnn), 8, 9a.

07D (decimal 0125)

Explanation: Module BLSFUP00 detected a permanent write error while writing a record to the problem directory.

System Action: The ABEND is preceded by message BLS04044I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 1, 2, 3, 5, 6a, 7c (for IPCSPRnn), 8, 9a.

07E (decimal 0126)

Explanation: Module BLSFUP00 detected an invalid DMCB (data access services control block) while writing a record to the problem directory.

System Action: The ABEND is preceded by message BLS04044I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

07F (decimal 0127)

Explanation: Module BLSFUP00 detected a permanent error while closing the problem directory.

System Action: The ABEND is preceded by message BLS04045I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 1, 2, 3, 5, 6a, 7c (for IPCSPRnn), 8, 9a.

080 (decimal 0128)

Explanation: A module detected an invalid DMCB (data access services control block) while attempting to close the problem directory.

System Action: The ABEND is preceded by informational message BLS04045I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

081 (decimal 0129)

Explanation: Internal error; Data Access Services received an invalid access request code.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

082 (decimal 0130)

Explanation: Internal error; Data Access Services detected, while attempting to locate the data set name parameter, that the IDCAMS DELETE model had been damaged.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

083 (decimal 0131)

Explanation: Internal error; Data Access Services enqueue validity check failed. The problem directory and data set directory resources are being opened in reverse order.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

084 (decimal 0132)

Explanation: A module detected an error while attempting to write a record into a data set. The messages that precede this ABEND specify in more detail the nature of the error.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

085 (decimal 0133)

Explanation: Internal error; module BLSCFDYN detected a return code from module BLSCANAL that was neither zero nor twelve.

System Action: Any existing messages on the message queue were routed to the terminal prior to the ABEND. The subcommand is terminated.

Programmer Response: Follow the programmer response(s) for any messages issued prior to the ABEND.

Problem Determination: Table I, items 4, 5, 8.

086 (decimal 0134)

Explanation: Internal error; Data Access Services cannot match any BLSUZZ2 control block to the current TCB.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

087 (decimal 0135)

Explanation: An internal error occurred while unchaining the DMCB (data access services control block) of the data set being freed from the DMCB chain. General purpose register 15 contains the reason code detailing the nature of the error:

Reason	Explanation
4	No entries exist on the DMCB chain.
8	An invalid DMCB or an invalid DMCB pointer was detected while searching the DMCB chain.
12	The DMCB being freed was not found to be an entry on the chain of DMCBs.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

089 (decimal 0137)

Explanation: A module detected an error while attempting to close a data set. Messages that precede this ABEND specify in more detail the nature of the error.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

08A (decimal 0138)

Explanation: A module detected an error while attempting to open a data set. Messages that precede this ABEND specify in more detail the nature of the error.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

08B (decimal 0139)

Explanation: Internal error; VSAM SHOWCB verb failed.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

08C (decimal 0140)

Explanation: An internally issued Data Access Services CLOSE request resulted in an error return code after VSAM OPEN verb had issued a warning return code.

System Action: The ABEND is preceded by informational message BLS03114I. The subcommand is terminated. Either the data set may not be closed or freed.

Programmer Response: None.

Problem Determination: If the data set is not properly closed, use IDCAMS VERIFY to close it. If the data set is not freed, issue a TSO FREE command. Table I, items 4, 5, 8.

08D (decimal 0141)

Explanation: An internal Data Access Services error caused a VSAM error.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

091 (decimal 0145)

Explanation: The area pointed to by the control block pointer specified in message BLS04072I does not contain the specified control block.

System Action: The ABEND is preceded by informational message BLS04072I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

093 (decimal 0147)

Explanation: Internal error; module BLSEADPR detected an invalid DMCB (data access services control block).

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

094 (decimal 0148)

Explanation: Internal error; module BLSEADP1 detected an invalid DMCB (data access services control block).

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

095 (decimal 0149)

Explanation: Internal error; a module received an unexpected return code.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

096 (decimal 0150)

Explanation: Internal error; unexpected messages were found on the message chain queue for either the problem directory or data set directory DMCB (data access services control block).

System Action: The message queue is left intact. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

097 (decimal 0151)

Explanation: Internal error; a nonzero return code was received from a conditional FREEMAIN macro instruction. All the requested storage could not be freed.

System Action: The ABEND is preceded by message BLS04014I. The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

098 (decimal 0152)

Explanation: Internal error; an invalid allocation model override parameter key was received.

System Action: The ABEND is preceded by message BLS03102I. IPCS processing is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

0C8 (decimal 0200)

Explanation: Internal error; module BLSEMDPR detected an invalid DMCB (data access services control block).

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

0C9 (decimal 0201)

Explanation: Internal error; module BLSEMDP1 detected an invalid DMCB (data access services control block).

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

3E7 (decimal 0999)

Explanation: A Data Access Services module is unable to obtain automatic storage from the DMCB (data access services control block) workarea.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

800 (decimal 2048)

Explanation: IKJPARS has placed an unsupported value in the IKJKEYWD PDE used for data types.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

801 (decimal 2049)

Explanation: BLSRM301 has received a boundary alignment designation of zero or greater than 1024K.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

802 (decimal 2050)

Explanation: BLSRM302 has received a field name of length zero or of length greater than 31 characters.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

803 (decimal 2051)

Explanation: BLSRMSGD has detected an unsupported data type code.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

804 (decimal 2052)

Explanation: BLSRACCL has detected an invalid parameter list.

System Action: The subcommand is terminated.

Programmer Response: None.

Problem Determination: Table I, items 4, 5, 8.

Problem Determination Table

Problem determination is the activity required to identify a failing hardware unit or program and determine who is responsible for support.

Problem determination is accomplished by using procedures specified by IBM. In some cases, these procedures may be initiated by a message or code which requires operator or programmer response. The response may include the requirement for additional problem-related data to be collected and will attempt, where possible, to indicate “probable” failure responsibility.

Problem determination information is included for applicable codes under the heading “Problem Determination.” Standard problem determination actions identified following the list of standard actions are identified as items of Tables I. Unique actions are identified following the list of standard actions to be taken. In any case, it is intended that the specified actions be taken before calling IBM for support.

Table I

If the problem recurs, follow the problem determination aids specified by the associated message or code before calling IBM for support:

1. If MSGLEVEL=(1,1) was not specified in the JOB statement, specify it and rerun the job.
2. Save the job stream associated with the job.
3. Save the system output (SYSOUT) associated with the job.
4. Make sure that the failing job step includes:
 - a. SYSABEND DD statement.
 - b. SYSUDUMP DD statement.
 - c. PL1DUMP DD statement.
 - d. SYSMDUMP DD statement.
5. Save the dump.
6. Execute the IEHLIST system utility program to obtain a list of the:
 - a. volume table of contents of the associated volume, specifying the FORMAT option.
 - b. Volume table of contents of the associated volume, specifying the DUMP option.
 - c. directory of the associated data set.
 - d. (Not applicable for MVS.)

7. Execute the IEBTPCH data set utility to print the:
 - a. directory of the applicable data set.
 - b. applicable data set.
 - c. applicable member.
 - d. applicable procedure.
8. Contact IBM for programming support.
9. Execute the Access Method Services LISTCAT command to:
 - a. list the contents of the applicable catalog.
 - b. list the catalog entries for the applicable objects and any related objects.

ABEND Cross Reference Table

This table presents a list of ABEND codes issued by IPCS modules. The list is sorted by ABEND code.

HEX (DEC)	Module Detecting	Module Issuing	HEX (DEC)	Module Detecting	Module Issuing
064 (0100)	BLSFGD00	BLSFGD00	081 (0129)	BLSCRQST	BLSCRQST
	BLSFGG00	BLSFGG00	082 (0130)	BLSCFAMS	BLSCFAMS
	BLSFGP00	BLSFGP00	083 (0131)	BLSCVALT	BLSCOPEN
065 (0101)	BLSFGD00	BLSFGD00	084 (0132)	BLSFREAS	BLSFREAS
	BLSFGG00	BLSFGG00	085 (0133)	BLSCFDYN	BLSCFDYN
	BLSFGP00	BLSFGP00	086 (0134)	BLSCRTCB	BLSCRTCB
066 (0102)	BLSFGD00	BLSFGD00	087 (0135)	BLSCFREE	BLSCFREE
	BLSFGG00	BLSFGG00	089 (0137)	BLSELPCT	BLSELPCT
	BLSFGP00	BLSFGP00		BLSELPDR	BLSELPDR
067 (0103)	BLSCGMF	BLSCGMF		BLSCFLOS	BLSCFLOS
068 (0104)	BLSDENQP	BLSDENQP	08A (0138)	BLSELPCT	BLSELPCT
069 (0105)	BLSDMSG0	BLSDMSG0		BLSELPDR	BLSELPDR
06A (0106)	BLSEDELO	BLSEDELO		BLSFOPEN	BLSFOPEN
06B (0107)	BLSEAUTH	BLSEAUTH	08B (0139)	BLSCRESE	BLSCRESE
06C (0108)	BLSEDELC	BLSEDELC		BLSCGETT	BLSCGETT
06D (0109)	BLSRESCK	BLSRESCK		BLSCPOIN	BLSCPOIN
06E (0110)	BLSCFAMS	BLSCFAMS		BLSCPUTT	BLSCPUTT
	BLSCFDYN	BLSCFDYN	08C (0140)	BLSCOPEN	BLSCOPEN
06F (0111)	BLSFOD00	BLSFOD00	08D (0141)	BLSCENDD	BLSCENDD
070 (0112)	BLSFOD00	BLSFOD00	091 (0145)	BLSCALOC	BLSCALOC
071 (0113)	BLSFUD00	BLSFUD00		BLSCRTCB	BLSCRTCB
072 (0114)	BLSUMONI	BLSUMONI		BLSDFPDR	BLSDFPDR
073 (0115)	BLSFDEL	BLSFDEL	093 (0147)	BLSEADPR	BLSEADPR
	BLSFDRES	BLSFDRES	094 (0148)	BLSEADP1	BLSEADP1
	BLSFUD00	BLSFUD00	095 (0149)	BLSFDALL	BLSFDALL
074 (0116)	BLSFUD00	BLSFUD00		BLSFDDDP	BLSFDDDP
075 (0117)	BLSFUD00	BLSFUD00		BLSFDD00	BLSFDD00
076 (0118)	BLSFUD00	BLSFUD00		BLSFDEL	BLSFDEL
077 (0119)	BLSFUD00	BLSFUD00		BLSFVRFY	BLSFVRFY
078 (0120)	BLSFUD00	BLSFUD00	096 (0150)	BLSFDDDP	BLSFDDDP
079 (0121)	BLSFUP00	BLSFUP00		BLSFDD00	BLSFDD00
07A (0122)	BLSELPFR	BLSELPFR	097 (0151)	BLSELPFM	BLSELPFM
	BLSFLONE	BLSFLONE	0C8 (0200)	BLSEMDPR	BLSEMDPR
07B (0123)	BLSFUP00	BLSFUP00	0C9 (0201)	BLSEMDP1	BLSEMDP1
	BLSELPDR	BLSELPDR	3E7 (0999)	BLSCAAMS	BLSCAAMS
	BLSELPFR	BLSELPFR		BLSCABLD	BLSCABLD
	BLSELOP	BLSELOP		BLSCADST	BLSCADST
	BLSFPREA	BLSFPREA		BLSCADYN	BLSCADYN
	BLSFPRES	BLSFPRES		BLSCAINT	BLSCAINT
	BLSFVRFY	BLSFVRFY		BLSCAMRE	BLSCAMRE
07C (0124)	BLSFUP00	BLSFUP00		BLSCANAL	BLSCANAL
07D (0125)	BLSFUP00	BLSFUP00		BLSCANLE	BLSCANLE
07E (0126)	BLSFUP00	BLSFUP00		BLSCCLSE	BLSCCLSE
07F (0127)	BLSFUP00	BLSFUP00		BLSCENDD	BLSCENDD
080 (0128)	BLSFUP00	BLSFUP00		BLSCRESE	BLSCRESE

HEX (DEC)	Module Detecting	Module Issuing
	BLSCFAMS	BLSCFAMS
	BLSCFDYN	BLSCFDYN
	BLSCFREE	BLSCFREE
	BLSCGETT	BLSCGETT
	BLSCGMF	BLSCGMF
	BLSCHOK	BLSCHOK
	BLSCOPEN	BLSCOPEN
	BLSCPOIN	BLSCPOIN
	BLSCPUTT	BLSCPUTT
	BLSCRFM	BLSCRFM
	BLSCRQST	BLSCRQST
	BLSCRTCB	BLSCRTCB
	BLSCSETT	BLSCSETT
	BLSCVALT	BLSCVALT

IPCS ABEND Codes (by Module)

This table lists ABEND codes issued by IPCS modules. It is sorted alphabetically by module name. It lists only those modules that issue ABEND codes.

Module	Code	Meaning of Code to Module
BLSCAAMS	999	More storage for DMCB work area unavailable.
BLSCABLD	999	More storage for DMCB work area unavailable.
BLSCADST	999	More storage for DMCB work area unavailable.
BLSCADYN	999	More storage for DMCB work area unavailable.
BLSCAINT	999	More storage for DMCB work area unavailable.
BLSCALOC	145	Pointer does not indicate specified control block.
BLSCAMRE	999	More storage for DMCB work area unavailable.
BLSCANAL	999	More storage for DMCB work area unavailable.
BLSCANLE	999	More storage for DMCB work area unavailable.
BLSCCLSE	999	More storage for DMCB work area unavailable.
BLSCENDDD	141	Data access error caused a VSAM error.
	999	More storage for DMCB work area unavailable.
BLSCRESE	139	Error in VSAM SHOWCB verb execution.
	999	More storage for DMCB work area unavailable.
BLSCFAMS	110	Invalid parameter passed to data access module.
	130	Data access detected a damaged IDCAMS DELETE model.
	999	More storage for DMCB work area unavailable.
BLSCFDYN	110	Invalid parameter passed to data access module.
	133	BLSCANAL return code neither 0 nor 12.
	999	More storage for DMCB work area unavailable.
BLSCFREE	135	Register 15 contains reason code for ABEND: 4 DMCB chain is empty. 8 DMCB or pointer invalid. 12 Specified DMCB not found in chain.
	999	More storage for DMCB work area unavailable.

Module	Code	Meaning of Code to Module
BLSCGETT	139	Error in VSAM SHOWCB verb execution.
	999	More storage for DMCB work area unavailable.
BLSCGMF	103	Unable to issue GETMAIN failure message, reasons for abend are in register 15 as follows: 20 No component table entry. 24 Zero component table entry. 28 No subcomponent table entry. 32 Zero subcomponent table entry. 36 No sub-subcomponent table entry. 40 Zero sub-subcomponent table entry. 60 Reserved storage area not large enough to contain the message. 64 The GETMAIN failure message routine does not permit inserts.
	999	More storage for DMCB work area unavailable.
BLSCHOK	999	More storage for DMCB work area unavailable.
BLSCOPEN	131	Error during OPEN enqueue validity check.
	140	VSAM OPEN verb issued a warning return code.
	999	More storage for DMCB work area unavailable.
BLSCPOIN	139	Error in VSAM SHOWCB execution.
	999	More storage for DMCB work area unavailable.
BLSCPUTT	139	Error in VSAM SHOWCB execution.
	999	More storage for DMCB work area unavailable.
BLSCRFM	999	More storage for DMCB work area unavailable.
BLSCRQST	129	Invalid access request code.
	999	More storage for DMCB work area unavailable.
BLSCRTCB	134	Unable to match BLSUZZ2 control block to TCB.
	145	Pointer does not indicate specified control block.
	999	More storage for DMCB work area unavailable.
BLSCSETT	999	More storage for DMCB work area unavailable.
BLSCVALT	131	Error during OPEN enqueue validity check.
	999	More storage for DMCB work area unavailable.
BLSDENQP	104	Resource error, directory already open.
BLSDFPDR	145	Pointer does not indicate specified control block.
BLSDMSG0	105	Module unable to produce message, reason for ABEND in register 15 as follows: 4-16 Return code from BLSUPUTN, BLSUPRTN, or BLSUTRMN. 20 No component table entry. 24 Zero component table entry. 28 No subcomponent table entry. 32 Zero subcomponent table entry. 36 No sub-subcomponent table entry. 40 Zero sub-subcomponent table entry. 48 Message insert caused message to exceed maximum length. 52 Message skeleton required more inserts than were supplied by the caller. 56 The caller supplied more inserts than were required by the skeleton. 60 GETMAIN returned a nonzero return code.

Module	Code	Meaning of Code to Module
BLSEADPR	147	Module detected an invalid DMCB.
BLSEADP1	148	Module detected an invalid DMCB.
BLSEAUTH	107	Module called with invalid authorization rule.
BLSEDELC	108	OPEN or CLOSE error for problem directory.
BLSEDEL0	106	I/O error occurred during a DELPROB data access.
BLSELPCT	137	Error detected during CLOSE of data set.
	138	Error detected during OPEN of data set.
BLSELPDR	123	Error detected during problem directory retrieval.
	137	Error detected during CLOSE of data set.
	138	Error detected during OPEN of data set.
BLSELPFM	151	Nonzero return code from FREEMAIN.
BLSELPFR	122	Insufficient storage available.
	123	Error during problem directory retrieval.
BLSELPOP	123	Error during problem directory retrieval.
BLSEMDPR	200	Module detected an invalid DMCB.
BLSEMDP1	201	Module detected an invalid DMCB.
BLSFCLOS	137	Error detected during CLOSE of data set.
BLSFDALL	149	Module received an unexpected return code.
BLSFDDDP	149	Module received an unexpected return code.
	150	Unexpected message found for data set or problem DMCB.
BLSFDD00	149	Module received an unexpected return code.
	150	Unexpected message found for data set or problem DMCB.
BLSFDEL	115	Error during data set directory retrieval.
	149	Module received an unexpected return code.
BLSFDRES	115	Error during data set directory retrieval.
BLSFREAS	132	Error during write to data set.
BLSFGD00	100	Error during OPEN for a directory.
	101	Error during a directory record retrieval.
	102	Error during CLOSE for a directory.
BLSFGG00	100	Error during OPEN for a directory.
	101	Error during a directory record retrieval.
	102	Error during CLOSE for a directory.
BLSFGP00	100	Error during OPEN for a directory.
	101	Error during a directory record retrieval.
	102	Error during CLOSE for a directory.
BLSFLONE	122	Insufficient storage found.
BLSFOD00	111	Record keys do not match.
	112	Record passed was not a base record.
BLSFOPEN	138	Error detected during OPEN of data set.
BLSFPREA	123	Error during problem directory retrieval.
BLSFPRES	123	Error during problem directory retrieval.

Module	Code	Meaning of Code to Module
BLSFUD00	113	DMCB found unrecoverable during OPEN.
	115	Error during data set directory retrieval.
	116	Invalid write request during WRITE to data set directory.
	117	Permanent error during WRITE to data set directory.
	118	Invalid DMCB found during WRITE to data set directory.
	119	Permanent error during CLOSE of data set directory.
	120	Invalid DMCB found during CLOSE of data set directory.
BLSFUP00	121	Invalid DMCB found during OPEN of data set.
	123	Error during problem directory retrieval.
	124	Invalid write request during WRITE to problem directory.
	125	Permanent error during WRITE to problem directory.
	126	Invalid DMCB found during WRITE to problem directory.
	127	Permanent error during CLOSE of problem directory.
	128	Invalid DMCB found during CLOSE of problem directory.
BLSFVRFY	123	Error during problem directory retrieval.
	149	Module received an unexpected return code.
BLSRESCK	109	Invalid equate symbol record found.
BLSUMONI	114	User-initiated diagnostic abend.

Error Recovery Hints

To assist in error recovery in IPCS, a special keyword is provided with IPCS subcommands.

The TEST keyword of IPCS subcommands places IPCS in a mode designed to support interactive testing of code that operates in the IPCS environment. It is not recommended for use for any other purpose.

If you anticipate an abnormal termination while testing a new exit routine written for the environment provided by the ASCBEXIT, TCBEXIT, or VERBEXIT subcommands and you wish to use TSO TEST facilities to isolate the cause of any problems, you should use the TEST keyword. When TEST is in effect, IPCS allows the TMP, the TSO TEST ESTAI functions, or both, to gain control when an abnormal termination occurs.

TEST mode also activates error-detection functions that have been developed to isolate dump data examination problems. Errors detected cause IPCS to abnormally terminate so that problems may be trapped close to the point of error.

The NOTEST keyword of IPCS subcommands places IPCS in the production mode of operation. This is the default. Automatic error recovery is attempted should errors occur in the IPCS environment.

When the NOTEST keyword is in effect, IPCS automatically recovers from most abnormal terminations without permitting TSO TEST to gain control.

ISPF Full Screen Dialog

The ISPF full-screen dump-viewing dialog operates in a unique environment, and makes use of facilities that are not used by the rest of IPCS, namely the ISPF table and variable services. Furthermore, the ISPF environment offers significant testing and debugging capabilities under option 7, Dialog Test. All of these facilities are available to the debugger to locate problems in the dialog.

The full screen dialog does not issue any ABEND codes. Messages issued are IPCS messages BLSMD00n, BLSMD01n, and BLSMD03n.

The return code is checked after each call from IPCS to an ISPF service, and if the code exceeds the error threshold for that service, the variable BLSFSER is set to the name of the failing service. The first byte of this variable being nonblank is used as the condition to bypass all further processing and go to the CLEANUP subroutine in BLSLDISP. In CLEANUP, when an error has occurred, the return code causing termination is translated to characters and put in variable BLSSRC. Panel BLSPFSER is displayed with message BLSMD000, in which both variables BLSFSER and BLSSRC are inserted by ISPF variable services.

To isolate problems associated with failing ISPF services, the following procedure is suggested:

1. Enter Option 7, Dialog Test, from the ISPF/PDF primary option menu.
2. Enter Dialog Test Option 8, Breakpoints, and set a breakpoint at the failing service.

3. Go to Dialog Test Option 1, Functions, and initiate the dialog by entering:

```
PGM==>BLSG  PARM==>PGM(BLSLDISP)
```

4. Stimulate the dialog to reproduce the error, noting the parameters used to invoke the failing service, as displayed on the breakpoint panel before the service is performed.
5. While stopped at the breakpoint, go to Dialog Test Option 3, Variables, to see the current values of all the function variables, they may provide a clue to the problem.
6. Enter GO on the breakpoint panel to proceed to the breakpoint after the service is performed/attempted, and note the return code. The return code value may be overtyped to a different value. The test facilities are also available at this time to examine the status of the variables or the tables.

AFT	ASN-first-table
ACB	Access method control block
ASCB	Address space control block
ASSB	Address space secondary block
ASMTV	Auxiliary storage manager vector table
AST	ASN-second-table
ASVT	Address space vector table
ASXB	Address space extension block
CDE	Contents directory entry
CQE	Console queue element
CSCB	Command scheduling control block
CSD	Common system data
CVT	Communications vector table
CVTXTNT2	OS/VS1-OS/VS2 common CVT extension
DCB	Data control block
GDA	Global data area
IOSB	I/O supervisor block
IRB	Interrupt request block
ISGGVT	Global resource serialization vector table
ISGGVTX	Global resource serialization vector table extension
ISGQCB	Global resource serialization queue control block
ISGQEL	Global resource serialization queue element
ISGQHT	Global resource serialization queue hash table
ISGRSV	Global resource serialization ring status vector
JESCT	Job entry subsystem control table
JSCB	Job scheduling control block
JQE	Job queue element
LCCA	Logical configuration communication area
LCCAVT	Logical configuration communication area vector table
LCH	Logical channel queue
LDA	Local data area
LPDE	Link pack directory entry
ORE	Operator reply element
OUCB	System resources manager user control block
OUSB	System resources manager user swappable block
OUXB	System resources manager user extension block
PART	Page address resolution table
PCB	Paging control block
PCCA	Physical configuration communication area
PCCAVT	Physical configuration communication area vector table
PFT	Page frame table
PFTE	Page frame table entry
PGT	Page table
PGTE	Page table entry
PRB	Program request block

PSA	Prefix storage area
PVT	Paging vector table
RMCT	Recovery management control table
RPL	Request parameter list
RQE	Request queue element
RSMHD	Real storage manager header
RTCT	Recovery termination control table
RTM2WA	Recovery termination manager 2 work area
SCVT	Secondary communications vector table
SGT	Segment table
SGTE	Segment table entry
SIRB	System interrupt block
SPQE	Subpool queue element
SRB	Service request block
SVRB	Supervisor call request block
TCB	Task control block
TIOT	Task I/O table
TIRB	Timer interrupt request block
TQE	Timer queue element
TSB	Terminal status block
UCB	Unit control block
UCBCTC	Unit control block, channel to channel adapter
UCBDA	Unit control block, direct access
UCBEXT	Unit control block extension
UCBGFX	Unit control block, graphics
UCBTAPE	Unit control block, tape
UCBTP	Unit control block, communications controller
UCBUR	Unit control block, unit record
UCB3270	Unit control block, 3270
UCM	Unit control module
UCME	Unit control module entry
WSAVTC	Work/save area vector table
WQE	Write-to-operator queue element
WSAVTG	Work/save area vector tables
XTLST	Extent list

Acronyms for IPCS Control Blocks

ARQ	Active record queue
BLSABDPL	Common exit parameter list
BLSARQ	Active record queue element
BLSBVT	Common data vector
BLSDMCB	Data access services
BLSDSD	Data set directory record map
BLSDVT	Central dat base vector
BLSFP	Facility parameter block
BLSLBLS	BLSLDISP dialog communication block
BLSNTRC	Intercept dialog block
BLSPDR	Problem directory record map
BLSQEXTP	BLSQECTI parameter list
BLSQSMPL	SUMMARY parameter list
BLSQSRA	Services router area
BLSRDATC	Data attribute descriptor
BLSRDATS	Address space descriptor
BLSRDATT	Data type descriptor
BLSRDTDI	Data type processing table
BLSRDUT	Dump table
BLSRESSY	Dump directory equate symbol record
BLSRFD	Find data
BLSRHSD	System/370 high speed dump record formats
BLSRPHSY	Dump directory program history record
BLSRPRX	Exit support data
BLSRRASY	Dump directory address space record
BLSRRBSY	Dump directory block list record
BLSRRCSY	Dump directory contents list record

Contains Restricted Materials of IBM
Licensed Materials — Property of IBM

BLSRRDSY	Dump directory data set record
BLSRSASY	Dump directory storage address record
BLSRSDSY	Dump directory session defaults
BLSRSST	Special symbol table
BLSRVALY	Task variable
BLSRVT	Debugging vector
BLSRZZ6	Dump file control block
BLSUVSFB	Dump directory
BLSUZZ1	Common area
BLSUZZ2	Active task variable
CBAT	Control block acronym table

General Acronyms for IPCS Control Blocks

DDIR	Dump directory
DMCB	Data access services control block
DSD	Data set directory
PDR	Problem directory
PDL	Parameter descriptor list
PSR	Problem status record



Appendix B. Common Module Characteristics

This section describes characteristics that are common to IPCS modules, and therefore are not included in individual module descriptions found in Section 3. The characteristics are register usage and patch labels.

IPCS Register Usage

This section has two parts: the first describes register usage during IPCS program linkage, and the second describes register usage internal to an IPCS program.

Program Linkage

When an IPCS program passes control to a separately-compiled IPCS program, the following linkage conventions are observed:

Regs	Use of Registers
0	Seldom used. It is used in the interface to several data access services programs. These programs return the address of a message chain in register 0.
1	Always employed as a parameter register. If only one storage address is required, register 1 contains that address. If two or more addresses are required, register 1 contains the address of a list of fullword addresses in storage.
2-12	Never employed.
13	The address of a 72-byte save area.
14	The return address.
15	The entry point address when a program is entered and a return code when it returns.

IPCS also contains numerous programs which are coded to comply with established system interface requirements: ESTAE and ESTAI exits, STAX exits, IKJPARS validity-checking exits, etc.

Internal Usage

Regs	Use of Registers
0-8	Used differently by each program.
9	Usually used to contain the address of the active task variable (see "BLSUZZ2" on page 406).
10	Used to address the parameter description list (PDL) generated by IKJPARS in most IPCS subcommands.
11	The address of the automatic storage (@DATD DSECT) in most programs.
12	The address of the code and program literal storage.
13	The address of a 72-byte save area. (Usually the contents of this register should match those of register 11.)
14-15	Used differently by each program.

Patch Labels

Two conventions are observed:

- Data access services programs contain a patch area labelled *DASPATCH*. It is initialized with binary zeros.
- Other IPCS programs contain a patch area labelled *BLRPATCH*. The initial bytes of the patch area are initialized with the literal CL8'ZAPAREA'. The remainder of the patch area consists of doublewords containing the EBCDIC program name.

Appendix C. IPCS Data Set Contention Protocols

IPCS components must contend for IPCS resources across address spaces in order to perform their functions. IPCS design specifies protocols that allow the sharing of resources while maintaining the integrity of the IPCS data base.

Data access services provides an interface between the IPCS subcommand and the IPCS data. Data access services handles all access to the problem directory and data set directory data sets for the problem and data management functions. For each data access request, the IPCS subcommand must supply the type of request it is making (for example, allocate, open, read/write, close, free), and the resource against which the request applies. It is the responsibility of the subcommand to minimize the amount of time the resource is used, even though it is data access services which actually obtains and frees the resource.

Allocating Resources

A data set resource is obtained for the subcommand making the request. Two reasons why the resource could possibly not be immediately obtained are:

1. The resource does not exist.
2. The resource is currently being exclusively used by another IPCS function or subcommand.

If the first condition exists, it is reflected back to the requesting subcommand. If the second condition exists, data access services waits for a short period of time (2 seconds) and attempts to obtain the resource again. Retry continues until the resource is obtained or until the user issues an attention interrupt, which is reflected back to the requesting subcommand.

Data access services creates a DMCB, locates the requested resource, and obtains/locks the resource via physical allocation disposition and/or ENQ. Whether shared or exclusive locking is used depends upon the resource requested, the contention protocol for the resource, and the indicated access mode to be used by the subcommand for the resource.

Data access services uses allocation models containing the data set default attributes. A model's contents include the default dynamic allocation parameter list, default ACB and RPL or DCB system access method control blocks, and an optional IDCAMS define cluster parameter list. These models are stored as part of the data access services program module.

Inter-Address Space Resource Contention

IPCS components contend for several resources. Among them are the problem directory (PDR), the data set directory (DSD), and data sets containing descriptions of problems.

Two similar resource sharing protocols are used for IPCS needs:

1. A high-contention protocol for access to the PDR and DSD data sets.
2. A low-contention protocol for access to all other data sets.

Data access services subroutines simplify compliance with these protocols.

High-Contention Protocol

The high-contention protocol is employed to serialize access to the PDR and DSD clusters.

1. If the subcommand requires access to the PDR and DSD clusters, the monitor function issues BLSCONAL requests which cause the clusters to be dynamically allocated using disposition SHR if they had not previously been allocated. Once allocated, the clusters remain allocated to the IPCS program until the program terminates.
2. When a transaction requires the use of a cluster, it issues a BLSOPEN specifying an access option of INPUT or OUTPUT. BLSOPEN function then does the following three things:
 - a. It validity checks the resource enqueue hierarchy for the PDR and DSD data sets. Ownership of the DSD data set resource by the requestor task when the PDR data set resource is requested will cause a task ABEND.
 - b. It issues a system ENQ using the following keywords:

QNAME = SYSBLSDS - Major Enqueue Resource Name
RNAME = Cluster Name - Minor Resource Name
E = exclusive (read/write) access to cluster
S = shared (read only) access to cluster
SYSTEMS = scope of request
RET=USE - type of request

If the resource is not obtained, the transaction enters an STIMER controlled wait loop that reissues the request. This permits the user to cause an attention interrupt to terminate the wait for the resource.

- c. Once BLSOPEN has obtained the ENQ resource, it issues a system OPEN for the cluster. If the transaction requires read-only access, the INPUT option is employed. If the transaction requires read/write access, the OUTPUT option is employed.

To handle the situation where a program outside the scope of IPCS accesses the PDR or DSD causing the OPEN to fail because of contention, an STIMER controlled wait loop is invoked in a similar manner to the ENQ wait loop.

3. Multiple data requests may be combined in one transaction, but no long-running computation, no terminal I/O, or other long-running I/O processes may be included.
 4. Immediately after the last data request has been handled, the transaction issues a BLSCLOSE request. The BLSCLOSE function does the following:
 - a. Issues a system CLOSE for the cluster.
 - b. Immediately after the cluster has been closed, a DEQ is issued to release the SYSTEM ENQ resource acquired above.
 5. During IPCS termination processing, the cluster is deallocated.

IPCS requires that VSAM SHAREOPTION(1) be used for the PDR and DSD clusters. This causes the MVS control program to enforce those portions of this protocol essential to the integrity of the clusters.

Low-Contention Protocol

The low-contention protocol is employed to serialize access to all data sets except the PDR and DSD.

1. When an application determines that access to a data set is required, that data set will be allocated. If read-only access is required, disposition SHR is used. If read/write access is required, disposition OLD or MOD is used.
2. Immediately after allocation has been completed, the data set is opened. If read-only access is required, the INPUT option is used. If read/write access is required, the OUTPUT option is used.
3. No restriction is applied to the number of data requests processed between OPEN and CLOSE.
4. Immediately after the last data request has been handled, the data set is closed.
5. Immediately after the data set has been closed, the data set is deallocated.

Index

A

abend codes
 by code 434-435
 by module 435-438
acronyms
 control blocks 441-442, 443
active task variable (ATV) 3
ADDDSN subcommand
 control flow 169
 method of operation 41
ADDPROB subcommand
 control flow 166
 method of operation 32
address resolution 11, 123, 124
 control flow 304
 method of operation 123-124
 module descriptions 304-310
address space mapping 11
 control flow 300
 method of operation 121-123
 module descriptions 299-303
allocation
 data set 89
 control flow 249
AMDPRFMT
 method of operation 64
analysis subcommands
 method of operation 55-64
 module descriptions 192-201
ASCB scanned by
 BLSVASCB 281
ASCBEXIT subcommand
 control flow 202
 method of operation 64
 module descriptions 203
ASMCHECK subcommand
 control flow 192
 method of operation 55
 module descriptions 192
ASSB scanned by
 BLSVASSB 281
AST scanned by
 BLSVASCB 281
ASTE scanned by
 BLSVASTE 281
ASVT scanned by
 BLSVASVT 281
ASXT scanned by
 BLSVASXB 281
ATV 3

B

block read routines
 module descriptions 282
BLS
 See BLSLBLS
BLSABDPL
 data area 326
BLSBVT
 data area 333
BLSCAAMS
 method of operation 89
 module description 250
BLSCABLD
 method of operation 89

 module description 251
BLSCADST
 module description 251
BLSCADYN 90
 method of operation 89
 module description 251
BLSCAINT
 method of operation 89
 module description 252
BLSCALOC 90
 control flow 249
 method of operation 89
 module description 252
BLSCAMER
 method of operation 89
 module description 254
BLSCAMIN
 module description 254
BLSCAMOD
 method of operation 89
 module description 255
BLSCAMPR
 module description 255
BLSCANAL
 method of operation 90
 module description 256
BLSCANLE
 module description 257
BLSCANLI
 module description 257
BLSCCLSE
 method of operation 92
 module description 257
BLSCENDD
 method of operation 92
 module description 258
BLSCERSE
 module description 258
BLSCFAMS
 method of operation 96
 module description 258
BLSCFDYN
 method of operation 96
 module description 259
BLSCFREE 97
 control flow 249
 method of operation 96
 module description 259
BLSCGETT
 method of operation 92
 module description 260
BLSCGMF
 module description 260
BLSCHOK
 module description 261
BLSCOPEN
 method of operation 92
 module description 261
BLSCPOIN
 method of operation 92
 module description 261
BLSCPUTT 93
 method of operation 92
 module description 261
BLSCRECV
 method of operation 92

module description 261
BLSCRESE
 method of operation 92
BLSCRFM
 module description 262
BLSCRQST 93
 control flow 250
 method of operation 92
 module description 262
BLSCRTCB
 module description 262
BLSCSETT 93
 method of operation 93
 module description 263
BLSCVALT
 method of operation 92
 module description 263
BLSDADSD 23, 148
 module description 263
BLSDAPDR 23, 148
 module description 263
BLSDC070
 module description 158
BLSDC128
 module description 158
BLSDC600
 module description 158
BLSDDEQP
 module description 172
BLSDDEQ0
 method of operation 92
BLSDDT00
 module description 158
BLSDENQP
 module description 173
BLSDENQ0
 method of operation 92
 module description 173
BLSDFDSD
 method of operation 18
 module description 263
BLSDFPDR
 method of operation 18
 module description 264
BLSDIFP0
 method of operation 16
 module description 136
BLSDIFP1 16
 method of operation 16
 module description 137
BLSDINI0
 method of operation 16
 module description 137
BLSDMCB
 data area 336
BLSDMSG5 10
 method of operation 99
 module description 272
BLSDMSG0 10, 100
 method of operation 99
 module description 273
BLSDPC00
 module description 158
BLSDSD
 data area 342
BLSDSTAE 136
 method of operation 17, 18
 module description 142
BLSDSV00
 module description 158

BLSDTIME
 module description 173
BLSDTM00
 module description 158
BLSDVECT 23
 module description 134
BLSDVT 23
 data area 343
BLSDWAIT
 module description 264
BLSEADPR
 control flow 166
 method of operation 32
 module description 174
BLSEADP1
 module description 174
BLSEAUTH
 module description 174
BLSEDEL
 module description 174
BLSEDEL0
 control flow 167
 method of operation 34
 module description 175
BLSEDEL1
 module description 175
BLSELPCC
 module description 175
BLSELPCL
 control flow 168
 module description 175
BLSELPCT
 module description 176
BLSELPDR
 module description 176
BLSELPDS
 module description 177
BLSELPFM
 module description 177
BLSELPFR
 module description 177
BLSELPPOP
 module description 178
BLSELP00
 control flow 168
 method of operation 36
 module description 178
BLSEMDPR
 control flow 168
 method of operation 39
 module description 178
BLSEMDP1
 module description 179
BLSFAD00
 control flow 169
 method of operation 41
 module description 179
BLSFAV00
 module description 179
BLSFBP00
 module description 180
BLSFCLOS
 module description 180
BLSFCN00
 module description 180
BLSFCONF
 module description 181
BLSFDALL

module description 181
BLSFDDDP
 method of operation 46
 module description 181
BLSFDD00
 control flow 170
 method of operation 45
 module description 181
BLSFDEL
 module description 182
BLSFDERS
 module description 182
BLSFDF00
 module description 183
BLSFDS00
 module description 183
BLSFFL00
 module description 184
BLSFFP00
 module description 184
BLSFGD00
 module description 184
BLSFGG00
 module description 185
BLSFGP00
 module description 185
BLSFLALL
 module description 185
BLSFLD00
 control flow 171
 method of operation 50
 module description 185
BLSFLONE
 module description 186
BLSFLP00
 module description 186
BLSFMD00
 method of operation 52
 module description 186
BLSFND00
 module description 187
BLSFND01
 module description 187
BLSFOD00
 module description 187
BLSFOPEN
 module description 188
BLSFP
 data area 345
BLSFPARS
 module description 188
BLSFPERA
 module description 189
BLSFPERS
 module description 189
BLSFPS00
 module description 189
BLSFREAS
 module description 183
BLSFSCRT
 module description 190
BLSFSD00
 module description 190
BLSFSL00
 module description 190
BLSFTL00
 module description 190
BLSFUD00
 module description 191
BLSFUP00

module description 191
BLSFVRFY
 module description 191
BLSG 3, 9
 control flow 239
 method of operation 83
 module description 239
BLSGINI2
 control flow 239
 module description 239
BLSGSCMD 3, 9
 control flow 240
 method of operation 83
 module description 240
BLSGX
 control flow 229
 method of operation 79
 module description 229
BLSHSD
 data area 378
BLSLBS
 data area 346
BLSLDISP 3, 9, 439
 control flow 245
 module description 246
BLSLDSTG
 control flow 245
 module description 247
BLSLDXCL
 module description 242
BLSLFINB
 module description 282
BLSLFIND
 module description 243
BLSLFISE
 control flow 245
 module description 247
BLSLIADS
 control flow 246
 module description 247
BLSLIADX
 control flow 246
 module description 248
BLSLIDSN
 control flow 246
 module description 248
BLSLMON 241
 module description 241
BLSLNTRC
 data area 352
BLSLPTRS
 control flow 246
 module description 248
BLSLRSET
 module description 243
BLSLSHOW
 module description 242
BLSLSNTX
 module description 242
BLSLSTRG
 module description 242
BLSPDR
 data area 357
BLSQALLO
 module description 206
BLSQCBAT
 method of operation 101
 module description 206

BLSQCFMT
control flow 210
method of operation 101, 102
module description 210

BLSQECT
control flow 208
method of operation 106
module description 208

BLSQESAR
control flow 212
method of operation 107
module description 212
module descriptions 212

BLSQESGU
control flow 211
method of operation 107
module description 211
module descriptions 211-212

BLSQEXTI
method of operation 101
module description 206

BLSQEXTP
data area 360

BLSQFORM
module description 211

BLSQFREE
module description 207

BLSQIFMT
method of operation 102
module description 211

BLSQROUT
method of operation 100
module description 205

BLSQSEL
control flow 209
method of operation 104

BLSQSIGR
module description 207

BLSQSMPL
data area 361

BLSQSRA
data area 363

BLSQSUM1
method of operation 63

BLSQSUM2
method of operation 63

BLSQSUM3
method of operation 63

BLSQSUM4
method of operation 63

BLSQSUM5
method of operation 63, 64

BLSQVPAS
module description 207

BLSRACC
module description 283

BLSRACCA 287
module description 282

BLSRACCB
module description 282

BLSRACCC 287
module description 282

BLSRACCI
method of operation 64
module description 284

BLSRACCL
method of operation 64, 70
module description 284

BLSRACCN
module description 283

BLSRACCP 287
module description 282

BLSRACCQ
module description 283

BLSRACCT 287, 288
module description 282

BLSRADDP
control flow 304
module description 305

BLSRADDR 124
control flow 304
method of operation 123
module description 305

BLSRADD5 124
control flow 304
method of operation 123
module description 306

BLSRADDT
control flow 304
module description 306

BLSRADDU
module description 306

BLSRADD1
control flow 304
module description 307

BLSRADD2
control flow 304
module description 307

BLSRADD3
control flow 304
module description 308

BLSRADD4
module description 308

BLSRADGP
module description 309

BLSRAMDX
module description 202

BLSRASCX
control flow 202
method of operation 64
module description 203

BLSRASM7
control flow 192
method of operation 55
module description 192

BLSRACOCL
module description 165

BLSRACOH
module description 165

BLSRACIN
module description 165

BLSRACOMK
control flow 193
method of operation 57
module description 193

BLSRACOML
module description 193

BLSRACOMP
control flow 223
method of operation 77
module description 223

BLSRACOPY
control flow 164
method of operation 29
module description 164

BLSRACOTA
module description 165

BLSRACPIC

- module description 223
- BLSRDATC**
 - data area 365
- BLSRDATK**
 - module description 309
- BLSRDATS**
 - data area 367
- BLSRDATT**
 - data area 369
- BLSRDTDT**
 - data area 370
- BLSRDTM1**
 - module description 134
- BLSRDTU1**
 - module description 134
- BLSRDUAL** 111, 285, 286
 - method of operation 109
 - module description 288
- BLSRDUBF**
 - module description 288
- BLSRDUCC** 285
 - module description 289
- BLSRDUCD** 285
 - module description 289
- BLSRDUCK** 109, 286
 - method of operation 109
 - module description 289
- BLSRDUCL** 286
 - module description 289
- BLSRDUDE**
 - method of operation 79
 - module description 231
- BLSRDUDR** 11, 109
 - method of operation 109
 - module description 231
- BLSRDUDT**
 - module description 231
- BLSRDUIC** 286
 - method of operation 110
 - module description 289
- BLSRDUIH** 286
 - method of operation 110
 - module description 290
- BLSRDUIN** 110, 285, 286
 - method of operation 110
 - module description 290
- BLSRDUIS** 286
 - method of operation 110
 - module description 290
- BLSRDUI2** 111, 285, 286
 - method of operation 111
 - module description 291
- BLSRDUI3** 111, 285, 287
 - method of operation 111
 - module description 291
- BLSRDULS**
 - method of operation 80
 - module description 231
- BLSRDUOP** 285, 287
 - method of operation 111
 - module description 291
- BLSRDUO2** 285, 287
 - method of operation 111
 - module description 291
- BLSRDUO3** 285, 287
 - method of operation 111
 - module description 291
- BLSRDUO4** 285, 287
 - method of operation 111
 - module description 292
- BLSRDUO5** 285, 288
 - method of operation 111
 - module description 292
- BLSRDUSC**
 - method of operation 13
 - module description 159
- BLSRDUSE** 288
 - module description 292
- BLSRDUSO** 148, 288
 - module description 292
- BLSRDUS1** 111, 288
 - method of operation 111
 - module description 292
- BLSRDUT**
 - data area 371
- BLSRDUT1** 285
 - module description 134
- BLSRDUT2** 285
 - module description 134
- BLSRDUT3** 285
 - module description 134
- BLSRDUT4** 285
 - module description 134
- BLSRDUT5** 285
 - module description 134
- BLSRDUVE** 288
 - module description 293
- BLSRDUVS** 288
 - module description 293
- BLSRECHA**
 - module description 316
- BLSREDEF**
 - control flow 224
 - method of operation 78
 - module description 224
- BLSREDUM**
 - control flow 225
 - method of operation 78
 - module description 225
- BLSREMAP**
 - control flow 226
 - method of operation 78
 - module description 226
- BLSRENPA**
 - method of operation 59
 - module description 194
- BLSRENPC**
 - method of operation 59
- BLSRENQA**
 - method of operation 59
 - module description 195
- BLSRENQK**
 - control flow 194
 - method of operation 59
 - module description 194
- BLSREQU**
 - control flow 236
 - method of operation 81
 - module description 237
- BLSRESAR**
 - module description 295, 314
- BLSRESCK**
 - module description 295
- BLSRESDR** 234
 - method of operation 80
 - module description 235
- BLSRESGA**
 - module description 295

BLSRESGC
 module description 295, 314
BLSRESGE 119
 method of operation 119
 module description 295
BLSRESGQ
 module description 295
BLSRESGU
 module description 295
BLSRESPU
 module description 295, 314
BLSRESRE
 control flow 238
 method of operation 81
 module description 238
BLSRESRN
 module description 238
BLSRESRU
 module description 235
BLSRESSA
 module description 295
BLSRESSR
 module description 295
BLSRESSS
 module description 295
BLSRESST
 control flow 237
 method of operation 81
 module description 237
BLSRESSY
 data area 373
 literal data 133
BLSRESYM
 control flow 227
 method of operation 78
 module description 227
BLSREVAL
 control flow 227
 method of operation 77
 module description 228
BLSRFD
 data area 376
BLSRFIND 213
 method of operation 66
 module description 213
BLSRFMOD
 control flow 217
 method of operation 67
 module description 217
BLSRFUCB
 control flow 218
 method of operation 67
 module description 218
BLSRIOSK
 control flow 203
 method of operation 64
 module description 203
BLSRLIST
 control flow 218
 method of operation 67
 module description 219
BLSRLSYM 235
 method of operation 80
 module description 236
BLSRLUCB
 control flow 219
 module description 219
BLSRMDSN
 module description 316
BLSRMSGA
 module description 317
BLSRMSGB
 module description 317
BLSRMSGD
 module description 318
BLSRMVSA
 module description 274
BLSRM077
 module description 195
BLSRM154
 module description 318
BLSRM185
 module description 318
BLSRM192
 module description 318
BLSRM193
 module description 319
BLSRM199
 module description 319
BLSRM200
 module description 319
BLSRM214
 module description 319
BLSRM300
 module description 320
BLSRM301
 module description 320
BLSRM302
 module description 320
BLSRPADS
 module description 156
BLSRPHAR
 module description 314
BLSRPHGE
 module description 314
BLSRPHSY
 data area 381
 literal data 133
BLSRPRX
 data area 382
BLSRPRXC
 module description 202
BLSRPUTT
 method of operation 64
BLSRRAAR
 control flow 300
 module description 300, 314
BLSRRACI
 control flow 300
 module description 301
BLSRRACR
 control flow 300
 method of operation 123
 module description 301
BLSRRACV
 control flow 300
 method of operation 123
 module description 302
BLSRRAGE
 control flow 300
 module description 302, 314
BLSRRASY
 data area 385
 literal data 133
BLSRRBGE
 module description 314
BLSRRBSY
 data area 387

literal data 133

BLSRRCGE
module description 314

BLSRRCSY
data area 388
literal data 133

BLSRRDAR
module description 314

BLSRRDGE
module description 314

BLSRRDPU
module description 314

BLSRRDSY
data area 389
literal data 133

BLSRRNCH
control flow 220
method of operation 68
module description 220

BLSRSAAR
method of operation 114
module description 275, 314

BLSRSADE
control flow 232
method of operation 81
module description 233

BLSRSAG
control flow 274
module description 275

BLSRSAGC
module description 276, 314

BLSRSAGE
module description 276

BLSRSAGN
module description 276

BLSRSAGP 114
method of operation 114
module description 277

BLSRSAGQ
method of operation 114
module description 277

BLSRSAGR
module description 278

BLSRSAGS
module description 278

BLSRSAGU
module description 279

BLSRSALI
control flow 232
method of operation 81
module description 233

BLSRSAPC
module description 279

BLSRSAPU
method of operation 114
module description 280, 314

BLSRSASY
data area 391
literal data 133

BLSRSCAN
control flow 233
method of operation 81
module description 234

BLSRSDAR
module description 314

BLSRSDGE
module description 314

BLSRSDSY
data area 394
literal data 134

BLSRSETD
control flow 161
method of operation 27
module description 161

BLSRSPIE
module description 303

BLSRSSC1 297

BLSRSSC2 297

BLSRSSC3 297

BLSRSSC4 297

BLSRSSC5 297

BLSRSSSA
module description 296

BLSRSSSC
module description 296

BLSRSSSP
module description 296

BLSRSST
data area 395

BLSRSST1
module description 135

BLSRSS01 297

BLSRSS02 297

BLSRSS03 297

BLSRSS04 297

BLSRSS05 297

BLSRSTAE
method of operation 18
module description 142

BLSRST00 196
method of operation 61
module description 196

BLSRST01
module description 196

BLSRSUMM
control flow 198
method of operation 63
module description 199

BLSRSUM1
control flow 198
module description 199

BLSRSUM2
control flow 198
module description 199

BLSRSUM3
control flow 198
module description 200

BLSRSUM4
control flow 198
module description 200

BLSRSUM5
control flow 198
module description 201

BLSRSUM6
control flow 198
module description 201

BLSRSYRB
module description 236

BLSRTCXB
control flow 204
method of operation 64
module description 204

BLSRVAL
module description 214

BLSRVALY
data area 396

BLSRVECT 23
module description 135

BLSRVEC2	finding data areas 298
module description 135	
BLSRVPAD	finding data areas 298
module description 158	
BLSRVPAS	finding data areas 298
module description 158	
BLSRVPCP	finding data areas 298
module description 158	
BLSRVPVA	finding data areas 298
module description 158	
BLSRVPVS	finding data areas 298
method of operation 14	
module description 158	
BLSRVRBX	finding data areas 299
control flow 204	
method of operation 64	
module description 205	
BLSRVSLD	finding data areas 299
control flow 159	
method of operation 14	
module description 159	
BLSRVT 23	finding data areas 299
data area 397	
BLSRZALC	finding data areas 299
module description 321	
BLSRZALN	finding data areas 299
module description 197	
BLSRZZ3R	finding data areas 299
module description 137	
BLSRZZ3U	finding data areas 299
module description 161	
BLSRZZ6	finding data areas 299
data area 400	
BLSRZZ6C	finding data areas 299
module description 293	
BLSR3270	finding data areas 299
control flow 221	
method of operation 21, 69	
module description 221	
BLSSAFT	finding data areas 298
finding data areas 298	
BLSSASCB	finding data areas 298
finding data areas 298	
BLSSASMV	finding data areas 298
finding data areas 298	
BLSSASTE	finding data areas 298
finding data areas 298	
BLSSASVT	finding data areas 298
finding data areas 298	
BLSSASXB	finding data areas 298
finding data areas 298	
BLSSCDE	finding data areas 298
finding data areas 298	
BLSSCOMM	finding data areas 298
finding data areas 298	
BLSSCSD	finding data areas 298
finding data areas 298	
BLSSCVT	finding data areas 298
finding data areas 298	
BLSSCVT2	finding data areas 298
finding data areas 298	
BLSSGDA	finding data areas 298
finding data areas 298	
BLSSGVT	finding data areas 298
finding data areas 298	
BLSSGVTX	finding data areas 298
finding data areas 298	
BLSSLCCA	finding data areas 298
finding data areas 298	
BLSSLCCV	finding data areas 298
finding data areas 298	
BLSSLLPA	finding data areas 298
finding data areas 298	
BLSSLPDE	finding data areas 298
finding data areas 298	
BLSSPCCA	finding data areas 298
finding data areas 298	
BLSSPCCV	finding data areas 298
finding data areas 298	
BLSSPFT	finding data areas 298
finding data areas 298	
BLSSPGTE	finding data areas 299
finding data areas 299	
BLSSPRIV	finding data areas 299
finding data areas 299	
BLSSPRIX	finding data areas 299
finding data areas 299	
BLSSPSA	finding data areas 299
finding data areas 299	
BLSSPVT	finding data areas 299
finding data areas 299	
BLSSQHT	finding data areas 299
finding data areas 299	
BLSSRSV	finding data areas 299
finding data areas 299	
BLSSRTCT	finding data areas 299
finding data areas 299	
BLSSSCVT	finding data areas 299
finding data areas 299	
BLSSSGTE	finding data areas 299
finding data areas 299	
BLSSTCB	finding data areas 299
finding data areas 299	
BLSSUCB	finding data areas 299
finding data areas 299	
BLSSUCM	finding data areas 299
finding data areas 299	
BLSSXTLS	finding data areas 299
finding data areas 299	
BLSTA	module description 316
module description 316	
BLSTB	module description 316
module description 316	
BLSTB4	module description 316
module description 316	
BLSTB5	module description 316
module description 316	
BLSTC	module description 316
module description 316	
BLSTF	module description 316
module description 316	
BLSTF2	module description 321
module description 321	
BLSTF4	module description 321
module description 321	
BLSTL	module description 316
module description 316	
BLSTM	module description 316
module description 316	
BLSTM4	module description 316
module description 316	
BLSTM5	module description 316
module description 316	
BLSTR	method of operation 131
method of operation 131	
module description 316	
BLSTU	module description 316
module description 316	

- BLSTU4
 - module description 316
- BLSTU5
 - module description 316
- BLSTVECT 23
- BLSTVT 23
- BLSTY
 - module description 316
- BLSTY1
 - module description 321
- BLSTY2
 - module description 321
- BLSTY3
 - module description 321
- BLSTY4
 - module description 321
- BLSTY4C3
 - module description 322
- BLSTY4C4
 - module description 322
- BLSTY8C1
 - module description 322
- BLSTY8C2
 - module description 322
- BLST01
 - method of operation 131
 - module description 315
- BLST02
 - method of operation 131
 - module description 316
- BLST03
 - module description 315
- BLST04
 - module description 315
- BLST04A
 - module description 315
- BLST04BL
 - module description 315
- BLST04BS
 - module description 315
- BLST04C
 - module description 315
- BLST04CR
 - module description 315
- BLST04CV
 - module description 315
- BLST04H
 - module description 315
- BLST05
 - method of operation 131
 - module description 322
- BLST06
 - method of operation 131
 - module description 316
- BLST08
 - method of operation 131
 - module description 315
- BLSU 8, 19
 - alias IPCS 14
 - control flow 136
 - method of operation 14
 - module description 137
- BLSUALLO 25
 - method of operation 24
 - module description 144
- BLSUBLDD
 - control flow 152
 - method of operation 25
 - module description 152
- BLSUBLDL
 - control flow 152
 - method of operation 23, 25
 - module description 152
- BLSUCLOS
 - control flow 230
 - module description 231
- BLSUDDSN
 - module description 294
- BLSUDYNA
 - module description 139
- BLSUFRE1
 - method of operation 25
 - module description 144
- BLSUGWDM
 - module description 145
- BLSUHELP
 - method of operation 24, 28
 - module description 160
- BLSUINI0
 - method of operation 16
 - module description 138
- BLSUINI1 8, 16
 - method of operation 16
 - module description 138
- BLSUINI2 8, 20
 - method of operation 17, 20
 - module description 139
- BLSUINI3 139
- BLSUINI8
 - module description 139
- BLSUINI9
 - method of operation 18
- BLSUIT01
 - method of operation 17
 - module description 324
- BLSUIT02
 - method of operation 17
 - module description 324
- BLSUIT03
 - method of operation 17
 - module description 324
- BLSUIT04
 - method of operation 17
 - module description 324
- BLSUIT05
 - method of operation 17
 - module description 324
- BLSULIN
 - control flow 228
 - method of operation 78
 - module description 228
- BLSUMDSN
 - module description 322
- BLSUMOAT 17
 - method of operation 16
 - module description 140
- BLSUMOII
 - module description 146
- BLSUMON 24
 - control flow 145
 - method of operation 20, 29, 83
 - module description 146
- BLSUMONA
 - method of operation 23
 - module description 146
- BLSUMONC 148
 - method of operation 23
 - module description 147

BLSUMONI 22
 control flow 145
 method of operation 20, 21
 module description 147

BLSUMONL 24
 method of operation 21, 22
 module description 148

BLSUMONT 148
 method of operation 23
 module description 148

BLSUMONX 148
 method of operation 23
 module description 149

BLSUMON2
 module description 149

BLSUMPKN
 module description 323

BLSUMPK1
 module description 323

BLSUMTOD
 module description 323

BLSUNOTE
 control flow 229
 method of operation 79
 module description 229

BLSUOPEN
 control flow 230
 module description 230

BLSUPARI
 method of operation 26
 module description 157

BLSUPARU
 module description 156

BLSUPGMC
 control flow 152
 method of operation 25
 module description 153

BLSUPGMD
 control flow 152
 method of operation 25
 module description 153

BLSUPGME
 method of operation 25
 module description 153

BLSUPGML
 control flow 152
 method of operation 25
 module description 154

BLSUPGMR
 control flow 152
 method of operation 25
 module description 154

BLSUPGMS
 method of operation 23, 25
 module description 154

BLSUPGMU
 method of operation 25
 module description 155

BLSUPRAB
 module description 140

BLSUPRCC
 module description 265

BLSUPRCL
 method of operation 18
 module description 141

BLSUPROP
 module description 141

BLSUPROX
 module description 141

BLSUPRTA
 method of operation 99
 module description 266

BLSUPRTN
 method of operation 100
 module description 266

BLSUPRTT
 method of operation 99
 module description 267

BLSUPUT 8, 133
 module description 133

BLSUPUTA
 module description 268

BLSUPUTC
 module description 268

BLSUPUTD
 module description 269

BLSUPUTI
 method of operation 64
 module description 269

BLSUPUTN
 method of operation 100
 module description 269

BLSUPUTO
 module description 270

BLSUPUTT
 method of operation 64
 module description 270

BLSUPUTV
 method of operation 99
 module description 270

BLSUSCM1
 method of operation 83
 module description 150

BLSUSIGR
 module description 156, 309

BLSUSTAE 19, 136
 method of operation 16, 18
 module description 142

BLSUSTAI 18, 136
 method of operation 17, 18
 module description 142

BLSUSTAS
 control flow 239
 module description 143

BLSUSTBL
 module description 135

BLSUSTDE
 module description 143

BLSUSTKD
 module description 150

BLSUSTKS
 module description 150

BLSUSTKT
 module description 150

BLSUTLCK
 module description 158

BLSUTLCL
 method of operation 18

BLSUTLOP 16
 method of operation 16

BLSUTRMA
 method of operation 99
 module description 271

BLSUTRMN
 method of operation 99, 100
 module description 272

BLSUTRMV
 module description 272

- BLSUTSO
 - method of operation 24, 28
 - module description 162
- BLSUVAST
 - module description 222
- BLSUVECT
 - module description 135
- BLSUVPS1
 - module description 158
- BLSUVPX1
 - module description 158
- BLSUVPX2
 - module description 158
- BLSUVP21
 - module description 158
- BLSUVP31
 - module description 158
- BLSUVP32
 - module description 158
- BLSUVP33
 - module description 158
- BLSUVP99
 - method of operation 13
 - module description 158
- BLSUVSAR
 - control flow 311
 - module description 311
- BLSUVSCA
 - module description 311
- BLSUVSCL 126
 - method of operation 18, 125
 - module description 312
- BLSUVSCR
 - module description 312
- BLSUVSEN
 - module description 310
- BLSUVSFB
 - data area 403
- BLSUVSGE
 - module description 310
- BLSUVSGU
 - module description 310
- BLSUVSMR
 - module description 312
- BLSUVSOP 16, 126
 - control flow 311
 - method of operation 16, 125
 - module description 312
- BLSUVSOQ 16
 - method of operation 16, 125
 - module description 313
- BLSUVSPO
 - module description 310
- BLSUVSPU
 - module description 310
- BLSUVSRE
 - module description 310
- BLSUVSWB
 - module description 310
- BLSUZZ1 3, 133
 - data area 404
 - modifiable data 133
- BLSUZZ2 3
 - data area 406
 - modifiable data 133
- BLSUZZ2C
 - method of operation 17
 - module description 141
- BLSUZZ2D 18
 - module description 141
- BLSUZZ2R 23
 - module description 151
- BLSVALA
 - module description 214
- BLSVALC
 - module description 215
- BLSVALF
 - module description 215
- BLSVALH
 - module description 216
- BLSVALP
 - module description 216
- BLSVALX
 - module description 216
- BLSVASCB
 - module description 281
- BLSVASSB
 - module description 281
- BLSVASTE
 - module description 281
- BLSVASVT
 - module description 281
- BLSVASXB
 - module description 281
- BLSVCDE
 - module description 281
- BLSVCQE
 - module description 281
- BLSVCSD
 - module description 281
- BLSVCVT
 - module description 281
- BLSVCVT2
 - module description 281
- BLSVGDA
 - module description 281
- BLSVGVT
 - module description 281
- BLSVGVTX
 - module description 281
- BLSVLCCA
 - module description 281
- BLSVLCCV
 - module description 281
- BLSVLDA
 - module description 281
- BLSVMOD
 - module description 281
- BLSVORE
 - module description 281
- BLSVPCCA
 - module description 281
- BLSVPCCV
 - module description 281
- BLSVPGTE
 - module description 281
- BLSVPSA
 - module description 281
- BLSVPVT
 - module description 281
- BLSVQCB
 - module description 281
- BLSVQEL
 - module description 281
- BLSVQHT
 - module description 281
- BLSVRB
 - module description 281
- BLSVRSV
 - module description 281

BLSVRTCT
 module description 281

BLSVSCVT
 module description 281

BLSVSGTE
 module description 281

BLSVTCB
 module description 281

BLSVUCB
 module description 281

BLSVUCM
 module description 281

BLSVWQE
 module description 281

BLSVXTLS
 module description 281

BLS9CMD1
 module description 162

BLS9INI2
 module description 162

BLS9MOI1
 method of operation 83
 module description 162

BLS9MONI
 module description 163

BLS9MPKN
 module description 323

BLS9PUT
 module description 135

BLS9STAE
 module description 143

BLS9STBL
 module description 135

BLS9STB1
 module description 135

Buffer Manager
 module descriptions 282

BVT
 See BLSBVT

C

CDE scanned by
 BLSVCDE 281

CDEMAJOR scanned by
 BLSVCDE 281

CDEMINOR scanned by
 BLSVCDE 281

CLIST support subcommands
 method of operation 77-79
 module descriptions 222

CLOSE subcommand
 control flow 230
 module descriptions 230-231

codes
 abend by code 434-435
 abend by module 435-438
 return 418-433

COMCHECK subcommand
 control flow 193
 method of operation 57
 module descriptions 192

common area (BLSUZZ1) 3

common exit services 10
 method of operation 100
 module descriptions 205

common module characteristics 445

common service functions 9
 method of operation 87-131

 module descriptions 249-324

COMPARE subcommand
 control flow 223
 method of operation 77
 module descriptions 223

components
 IPCS 6-9

control block formatting service
 method of operation 101

control blocks
 acronym definitions 441-442, 443
 descriptions 325-414
 overview 4

COPYDUMP subcommand
 control flow 164
 method of operation 29-31
 module description 164-166

CQE scanned by
 BLSVCQE 281

CSD scanned by
 BLSVCSD 281

CVT scanned by
 BLSVCVT 281

CVTXTNT2 scanned by
 BLSVCVT2 281

D

data access services 9
 method of operation 87
 module descriptions 249-265
 overview 87

data areas
 BLSABDPL 326
 BLSBVT 333
 BLSDMCB 336
 BLSDSD 342
 BLSDVT 343
 BLSFP 345
 BLSLBLS 346
 BLSLNTRC 352
 BLSPDR 357
 BLSQEXTP 360
 BLSQSMPL 361
 BLSQSRA 363
 BLSRDATC 365
 BLSRDATS 367
 BLSRDATT 369
 BLSRDTDT 370
 BLSRDUT 371
 BLSRESSY 373
 BLSRFD 376
 BLSRHSD 378
 BLSRPHSY 381
 BLSRPRX 382
 BLSRRASY 385
 BLSRRBSY 387
 BLSRRCSY 388
 BLSRRDSY 389
 BLSRSASY 391
 BLSRSDSY 394
 BLSRSST 395
 BLSRVALY 396
 BLSRVT 397
 BLSRZZ6 400
 BLSUVSFB 403
 BLSUZZ1 404

BLSUZZ2 406
data set allocation
 control flow 249
 method of operation 89
data set contention 447
data set deallocation
 method of operation 96
data set description 4
data structures 4
deallocation
 data set 96
DELDSN subcommand
 control flow 170
DELPROB subcommand
 connection to DELDSN function 46
 control flow 167
 method of operation 34
dialog task variable (DTV) 3
DMCB
 See BLSDMCB
DROPDUMP subcommand
 method of operation 79
 module descriptions 231
DROPMAP subcommand
 control flow 232
 module descriptions 233
DROPSYM subcommand
 control flow 234
 method of operation 80-81
 module descriptions 235-236
DSDREC
 See BLSDSRD
DSPL3270 subcommand 21
 control flow 221
 method of operation 21
 module descriptions 220
DTDT
 See BLSRDTDT
DTV 3
dump access 10
 method of operation 108-109
 module descriptions 282-284
dump data formatting 12-131
 module descriptions 315-324
dump directory handling 11-12
 control flow 311
 method of operation 125
 module descriptions 310-313
dump directory handling subcommands
 method of operation 79-82
 module descriptions 230-238
dump directory properties 128
dump directory record management
 module descriptions 313-315
dump directory records 127-128
 See also ES, PH, RA, RB, RC, RD, SA, SD
dump display reporter
 control flow 241
dump initialization 11
 control flow 285-288
 method of operation 109-111
 module descriptions 285-294
DUT
 See BLSRDUT
DVT 23
 See also BLSDVT

E

ECT exit service
 method of operation 106
 module descriptions 208
END subcommand
 method of operation 29
ENQCHECK subcommand
 control flow 194
 method of operation 59
 module descriptions 194-195
EQUATE subcommand
 control flow 236
 method of operation 81
 module descriptions 237
equate symbol service
 control flow 212
 method of operation 107
 module descriptions 212
ES 127
 See also BLSRESSY
EVALDEF subcommand
 control flow 224
 method of operation 78
 module descriptions 224
EVALDUMP subcommand
 control flow 225
 method of operation 78
 module descriptions 224-225
EVALMAP subcommand
 control flow 226
 method of operation 78
 module descriptions 225-226
EVALSYM subcommand
 control flow 227
 method of operation 78
 module descriptions 226-227
EVALUATE subcommand
 control flow 227
 method of operation 77
 module descriptions 228
exit interface services
 module descriptions 202
exit services router service
 method of operation 100
exit support subcommands
 module descriptions 202-212

F

find routines 11
FIND subcommand
 control flow 213
 method of operation 66
finding data areas
 control flow 297
 method of operation 119-121
 module descriptions 297-299
 tutorial 119-121
FINDMOD subcommand
 control flow 217
 method of operation 67
FINDUCB subcommand
 control flow 218
 method of operation 67
format model processing
 method of operation 102
FPBLOK
 See BLSFP

- full screen dump viewing
 - control flow 221
 - method of operation 69-71
 - module descriptions 220
- full screen dump viewing ISPF dialog
 - control flow 245-246
 - module descriptions 244-249
- full screen initialization
 - method of operation 69-70
- full screen termination 71
- full-screen dump viewing ISPF dialog
 - method of operation 83

G

- GDA scanned by
 - BLSVGDA 281
- get symbol service
 - control flow 211
 - method of operation 107
 - module descriptions 211

H

- HELP command
 - method of operation 28
- HELP subcommand
 - module description 160
- HSD
 - See BLSRHSD

I

- initialization/termination 3
 - module descriptions 136-143
- initialization, IPCS
 - method of operation 16-17
- initialization, termination
 - control flow 136
- INTEGER subcommand
 - control flow 228
 - method of operation 78
 - module descriptions 228
- intertask communication 3
- IOSCHECK subcommand
 - control flow 203
 - method of operation 64
 - module description 203
- IPCS command
 - alias of module BLSU 137
 - control flow 136
 - method of operation 14
 - module description 137
- IPCS STAE exits
 - module description 142
- IPCS/ISPF environment
 - control flow 239
- IPCSDDIR command
 - control flow 159
 - method of operation 14
 - module description 159
- IRB scanned by
 - BLSVRB 281
- ISGGVT scanned by
 - BLSVGVT 281
- ISGGVTX scanned by
 - BLSVGVTX 281

- ISGQCB scanned by
 - BLSVQCB 281
- ISGOEL scanned by
 - BLSVQEL 281
- ISGQHT scanned by
 - BLSVQHT 281
- ISGRSV scanned by
 - BLSVRSV 281
- ISPEXEC subcommand
 - control flow 229
 - method of operation 79
 - module descriptions 229
- ISPF dialog programs
 - introduction 9
 - module descriptions 239-249
- ISPF logical screen task 3-4

L

- LCCA scanned by
 - BLSVLCCA 281
- LCCAVT scanned by
 - BLSVLCCV 281
- LDA scanned by
 - BLSVLDA 281
- line-oriented dump viewing
 - method of operation 66-71
 - module descriptions 213-220
- LIST subcommand
 - control flow 218
 - method of operation 67-68
- LISTDSN subcommand
 - control flow 171
 - method of operation 50
- LISTDUMP subcommand
 - method of operation 80
 - module descriptions 231
- LISTMAP subcommand
 - control flow 232
 - module descriptions 233-234
- LISTPROB subcommand
 - control flow 168
 - method of operation 36
- LISTSYM subcommand
 - control flow 235
 - method of operation 80-81
 - module descriptions 235-236
- LISTUCB subcommand
 - control flow 219
- LPDE scanned by
 - BLSVCDE 281
- LPDEMAJOR scanned by
 - BLSVCDE 281
- LPDEMINOR scanned by
 - BLSVCDE 281

M

- master task 3
- master task variable (MTV) 3
- message build function 10
 - method of operation 99
 - module descriptions 272
- message handling 98
- message routing routines 268-271
- MODDSN subcommand

control flow 172
method of operation 52
MODPROB subcommand
control flow 168
method of operation 39
module(program name) scanned by
BLSVMOD 281
monitor 8
control flow 145
method of operation 19-26
module descriptions 143-155
MTV 3

N

non-executable modules 133
NOTE subcommand
control flow 229
method of operation 79
module descriptions 229

O

OPEN subcommand
control flow 230
module descriptions 230
operating system
relation 1
ORE scanned by
BLSVORE 281

P

parse validity checking
method of operation 82
module descriptions 157-158
patch labels 446
PCCA scanned by
BLSVPCCA 281
PCCAVT scanned by
BLSVPCCV 281
PDREC
See BLSPDR
PGT scanned by
BLSVPGTE 281
PGTE scanned by
BLSVPGTE 281
PH 127
See also BLSRPHSY
PRB scanned by
BLSVRB 281
PRDMP exit services
module descriptions 283
PRDMP exit support subcommands
method of operation 64-65
print and terminal services 10
method of operation 98-100
module descriptions 265-272
print routines 265-267
problem and data management subcommands
method of operation 32
module descriptions 166-191
processing subtask 3
processing task variable (PTV) 3
program management
control flow 152
method of operation 25-26

module descriptions 152-155
PSA scanned by
BLSVPSA 281
PTV 3
PVT scanned by
BLSVPVT 281

R

RA 127
See also BLSRRASY
RB 127
See also BLSRRBSY
RB scanned by
BLSVRB 281
RC 127
See also BLSRRCSY
RD 128
See also BLSRRDSY
records
dump directory 127-128, 133-134
register conventions 445
RENUM subcommand
control flow 238
method of operation 81
module descriptions 238
response processing 70
routing routines
method of operation 99
RTCT scanned by
BLSVRTCT 281
RUNCHAIN subcommand
control flow 220
method of operation 68-69
RVT 23
See also BLSRVT

S

SA 128
See also BLSRSASY
scan routines 11
control flow 280
method of operation 114-116
module descriptions 280-281
tutorial 114-116
SCAN subcommand
control flow 233
module descriptions 234
screen image generation 70
SCVT scanned by
BLSVSCVT 281
SD 128
See also BLSRSDSY
select ASID service
method of operation 104
module descriptions 209
session control subcommands
module descriptions 160-166
SETDEF command
method of operation 28
SETDEF subcommand
control flow 161
method of operation 27
module description 161
SGT scanned by

BLSVSGTE 281
SGTE scanned by
 BLSVSGTE 281
SIRB scanned by
 BLSVRB 281
special symbols, processing 296-297
SST
 See BLSRSST
STACK subcommand
 control flow 237
 method of operation 81
 module descriptions 237
STATUS subcommand
 control flow 196
 method of operation 61
 module descriptions 195-197
storage access routines
 module descriptions 283
storage management 25
 method of operation 24
storage map management 11
 control flow 274
 method of operation 112-114
 module descriptions 273
subcommand processors
 introduction 8
 module descriptions 155-238
subpool usage 25
SUMMARY subcommand
 control flow 198
 method of operation 63-64
 module descriptions 198-201
SVRB scanned by
 BLSVRB 281
symbol management 11
 method of operation 116-118
 module descriptions 294-295
SYSDSCAN command
 method of operation 13-14
 module descriptions 159

T

task structure 1-4
task variable (BLSUZZ2) 3
TASKLIB keyword 16
TCB scanned by
 BLSVTCB 281
TCBEXIT subcommand
 control flow 204
 method of operation 64
 module descriptions 204
terminal attention processing 21
terminal transmission routines 271-272
termination
 full screen 71
 master task 18-19
 processing subtask 18
TEST keyword 439
TIRB scanned by
 BLSVRB 281
transmit and control routines
 method of operation 99

TSO command
 method of operation 28
 module description 162-163
TSO utility commands
 method of operation 13-14
TVT 23

U

UCB scanned by
 BLSVUCB 281
UCBCTC scanned by
 BLSVUCB 281
UCBDA scanned by
 BLSVUCB 281
UCBGFX scanned by
 BLSVUCB 281
UCBTAPE scanned by
 BLSVUCB 281
UCBTP scanned by
 BLSVUCB 281
UCBUR scanned by
 BLSVUCB 281
UCB3270 scanned by
 BLSVUCB 281
UCM scanned by
 BLSVUCM 281

V

value resolution service
 module descriptions 214
Vector Facility 61
VERBEXIT subcommand
 control flow 204
 method of operation 65
 module descriptions 205

W

WQE scanned by
 BLSVWQE 281

X

XTLST scanned by
 BLSVXTLS 281

Z

ZZ1
 See BLSUZZ1
ZZ2
 See BLSUZZ2
ZZ4
 See BLSUVSFB
ZZ6
 See BLSRZZ6



"Restricted Materials of IBM"

All Rights Reserved

Licensed Materials - Property of IBM

©Copyright IBM Corp. 1982, 1986

LY28-1298-2

S370-37



Printed in U.S.A.

LY28-1298-2

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

What is your occupation? _____

How do you use this publication? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the

"Restricted Materials of IBM"

All Rights Reserved

Licensed Materials - Property of IBM

(Except for Customer-Originated Materials)

©Copyright IBM Corp. 1982, 1986

LY28-1298-2

S370-37

Reader's Comment Form

Cut or Fold Along Line

Fold and tape

Please Do Not Staple

Fold and tape



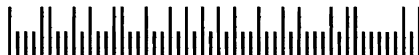
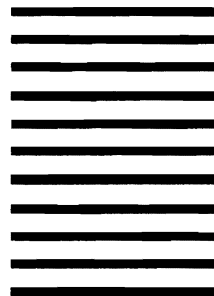
NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO Box 390
Poughkeepsie, New York 12602



Fold and tape

Please Do Not Staple

Fold and tape

Printed in U.S.A.

IBM®

LY28-1298-2

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

What is your occupation? _____

How do you use this publication? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the

MVS/Extended Architecture Interactive Problem Control System Logic and Diagnosis

"Restricted Materials of IBM"

All Rights Reserved

Licensed Materials - Property of IBM

(Except for Customer-Originated Materials)

©Copyright IBM Corp. 1982, 1986

LY28-1298-2

S370-37

Reader's Comment Form

Cut or Fold Along Line

Fold and tape

Please Do Not Staple

Fold and tape



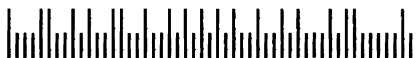
NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO Box 390
Poughkeepsie, New York 12602



Fold and tape

Please Do Not Staple

Fold and tape

IBM®

Printed in U.S.A.

MVS/Extended Architecture
Interactive Problem
Control System
Logic and Diagnosis

READER'S
COMMENT
FORM

LY28-1298-2

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

What is your occupation? _____

How do you use this publication? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the

"Restricted Materials of IBM"

All Rights Reserved

Licensed Materials - Property of IBM

(Except for Customer-Originated Materials)

©Copyright IBM Corp. 1982, 1986

LY28-1298-2

S370-37

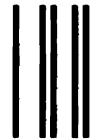
Reader's Comment Form

Cut or Fold Along Line

Fold and tape

Please Do Not Staple

Fold and tape



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO Box 390
Poughkeepsie, New York 12602



Fold and tape

Please Do Not Staple

Fold and tape

IBM®

Printed in U.S.A.

LY28-1298-02

